Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Departamento de Física

Intelligent Sensing Anywhere

Remote Vital Signs Monitoring

Data Processing and Transmission Module

by

Nuno Miguel Silva Varelas

A thesis submitted for the degree of

Master of Biomedical Engineering

Coimbra, September 2008

ABSTRACT

One of today's major society concerns in the medical field is the ageing of population. Over the last decades, pressure in healthcare providers has been increasing due to not only the high incidence of chronic diseases but also the constant rise in healthcare demands. Under this healthcare providers' pressure emerges a gap that allows the insertion of advanced medical technologies that will not only relieve healthcare provider's pressure but also improve quality of care.

The aim of this project is to develop a system's prototype whose function is to provide vital signs' remote monitoring in both hospital and Telehomecare environments in order to allow vital data access by a clinician in real-time and anywhere.

The system includes wireless and wearable sensors which acquire and process several vital signs and transmit them into a nearby Transmission Module over Bluetooth technology. Data are then transmitted into a remote Datacenter in order to allow access over a web-based server or a nursing central by a clinician. Wi-Fi, Ethernet or GSM/GPRS are the three remote communication technologies that allow integration in both hospital and Telehomecare environment.

Throughout this thesis it will be analyzed the project requirements and it will be described a personalized Remote Vital Signs Monitoring solution which includes the integration of a wireless Bluetooth oximeter and data remote transmitting over Ethernet. The description includes entire system specifications and Transmission Module's firmware design. Additionally some tests performed in the final stage of the project indicated that the developed system properly performs the required functions.

RESUMO

Uma das preocupações mais patentes na sociedade de hoje relativamente à área da saúde é o envelhecimento da população. A pressão nas unidades de saúde tem vindo continuamente a acentuar-se na última década e tem como causas não só a elevada incidência de doenças crónicas mas também o contínuo aumento da exigência nos cuidados de saúde. Sob esta pressão surge uma oportunidade de inserção de tecnologias médicas avançadas que irão não só aliviar a pressão nos prestadores de cuidados de saúde mas também melhorar a qualidade destes.

O objectivo deste projecto é desenvolver um protótipo de um sistema cuja função é proporcionar uma monitorização remota de sinais vitais tanto em ambiente hospitalar como domiciliário de forma a permitir aos profissionais clínicos o acesso aos dados dos sinais vitais em tempo-real e em qualquer parte.

O sistema é constituído por sensores sem fios, que possam facilmente ser usados pelo paciente e que processam e transmitem por Bluetooth os sinais vitais para um Módulo Concentrador localizado nas imediações. Os dados são depois transmitidos remotamente para um Módulo Terminal onde são armazenados de forma a permitir o acesso a profissionais clínicos via um servidor Web ou uma central de enfermagem. As três tecnologias de comunicação remota usadas são Wi-Fi, Ethernet ou GSM/GPRS de forma a levar a cabo a integração do sistema tanto em ambiente hospitalar como domiciliário.

No decurso desta tese vão ser analisados os requisitos do projecto e descrita uma solução personalizada de Monitorização Remota de Sinais Vitais que, por sua vez, inclui a integração de um oxímetro Bluetooth e transmissão de dados remota via Ethernet. A descrição do sistema inclui toda a especificação deste e a elaboração do firmware do Módulo Concentrador. Alguns testes realizados na fase final do projecto permitiram obter resultados que indicam que o sistema desenvolvido realiza correctamente as funções exigidas.

ACKNOWLEDGMENTS

First, I would like to thank to the Project coordinators, Professors Carlos Correia and José Basílio Simões for making this project possible.

To my supervisor Paulo Santos that, despite his filled schedule, managed to find a little time to contribute to this project. My special thanks for his support and advice.

I also want to thank Engineers Miguel Aragão, Artur Vieira, Tiago Marçal and all the other ISA Engineers that, for their patience and availability, helped solving the many problems found in the way.

I would like to thank Engineers Nuno Alves and Miguel Cardoso for having strongly contributed to my apprenticeship in the field of electronics and microcontrollers. Without their support it wouldn't be possible to reach this conclusion stage of the project.

I'm very grateful to Engineers Catarina Pereira and José Luis Malaquias for their help and interest in the project, always with good suggestions and new opportunities for the project.

To all ISA workers that, for their sympathy and constant good mood, made pleasant the time passed at ISA.

I'm thankful to my project colleagues for their constant support, dedication and friendship and for never stopping believing that it was possible to achieve a good result.

To all my friends that contributed, in a way or another, for the accomplishment of this work, whether by their companionship and positive state of mind along the several hours of work or by simply existing. Thank you.

Finally, a very special thanks to all my family and to Sara, for being special and for never leaving me in this crucial stage of my academic life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND DEFINITIONS

A/D	Analog-to-Digital
ACK	Acknowledge
ADC	Analog-to-Digital Converter
ANSI	American National Standards Institute
BCD	Binary-Coded Decimal
BRG	Baud Rate Generator
CEI	Electronics and Instrumentation Centre
CPU	Central Processing Unit
CS	Chip Select
DHCP	Dynamic Host Configuration Protocol
ECG	Electrocardiogram
EEPROM	Electrically Erasable Programmable Read-Only Memory
Flash	Non-volatile memory
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HCM	Host Controlled Mode
I/O	Input/output
I_2C	Inter Integrated Circuit
ICD	In-Circuit Debugger
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISA	Intelligent Sensing Anywhere
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
LED	Light Emitting Diode
LSB	Least Significant Bit
MAC	Media Access Control
microSD card	micro Secure Digital Card
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
OSI	Open Systems Interconnection
PC	Personal Computer
PDA	Personal Digital Assistant
PDP	Programmed Data Processor
PLL	Phase-Locked Loop

Plug and Play	Computer feature that allows the addition of a new device without requiring reconfiguration
R&D	Research and Development
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RS-232	Recommended Standard 232
RS-485	Recommended Standard 485
RTC	Real Time Clock
SCK	SPI Serial Clock
SCL	I_2C Serial Clock
SDA	I_2C Serial Data
SDI	Serial Data Input
SDO	Serial Data Output
SMS	Short Message Service
SPI	Serial Peripheral Interface
SPICE	Simulated Program with Integrated Circuits Emphasis
SpO2	Saturation of Peripheral Oxygen
SS	Slave Select
SSP	Secure Simple Pairing
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
Unix	Computer operating system
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VSM	Virtual System Modelling
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WPAN	Wireless Personal Area Network

CHAPTER 1

1. Introduction

1.1. Motivation

One of the most important achievements in today's modern society is the increase in life expectancy. In Portugal, one of the worst case scenarios in Europe, life expectancy approximately increased by ten years between 1975 and 2005, reaching 81 years for women and 74.5 years for men. In addition, projections suggest life expectancy continues to increase reaching 84.7 and 79 years in 2050 for women and men respectively (1).

Table 1.1 : Life expectancy at birth in years, 1975-2050, Portugal (1).

	1975	1980	1985	1990	1995	2000	2005	2050
Women	72.1	74.8	76.4	77.5	79.0	79.9	81.0	84.7
Men	64.7	67.8	69.4	70.6	71.8	72.9	74.5	79.0

However, life expectancy increase is one of the main causes of an important concern in today's modern societies which is ageing of population. Over the last decades, the number of elderly people has been rising to alarming numbers (1).

Elderly people are the group of population in which there is a bigger incidence in chronic diseases. Due to the constant rise in healthcare demands and to the high incidence of chronic diseases which require periodic treatment, pressure in healthcare providers has been increasing. There is a high determination to increase life expectancy though, at the same time, healthcare providers want to reduce pressure caused by the increase in chronic diseases and healthcare demands. It is a vicious circle as attached to the increase in life expectancy is the population ageing which, in turn, causes chronic diseases incidence to rise and therefore pressure in healthcare providers continues to get even bigger (2; 3).

Pressure in healthcare providers arises under several forms. These include costs with human resources and newer and more expensive treatments as well as the infrastructural saturation due to high patients' affluence (2).

Vital Signs Remote Monitoring concept appears as a solution that can minimize some of the referred issues, especially on the related chronic diseases scenarios. Additionally, devices that enable remote monitoring concept can be used as a supplement to the existent methods thus providing advances in patients' healthcare.(4)

Vital Signs Remote Monitoring systems can be inserted into two distinct scenarios. The first is Telehomecare which enables clinicians to access patient's health data over long distances, allowing patients to be at their home while being followed by the clinician. The other scenario is hospital environment Remote Monitoring which offers clinicians the ability to monitor their patients remotely whether through a central station monitor or a portable handheld device such as a PDA. Both presented scenarios support the idea of Telehealth, that is, the use of electronics and communication technologies in order to enable remote health care (2; 4).

Telehomecare allows patients with reduced mobility or access constraints to central healthcare units to get access to health care. In addition, it releases pressure in healthcare units because it allows patients in terminal treatment phases to go home and be followed by their caregivers. Furthermore, it constitutes another means of diagnosis, especially to people with chronic diseases (2).

On the other hand, hospital environment Remote Monitoring releases pressure on the caregivers because it centralizes patients' monitoring data, thus requiring less care giving personnel. It can also offer an improved alarm management system, thus resulting in shorter response times. Wireless capabilities can provide a comfortable monitoring environment with no wires and wider mobility to the patient. Digital data storage of the patients' vital signs data allows posterior historic access in order to detect possible abnormality causes or to obtain a simple diagnosis (2).

Benefits for both general population and healthcare providers, which can be introduced by the implementation of Telehealth systems, constitute a big amount of motivation for the current project. On the other hand, technology evolution constitutes a big stimulus to the development of a Vital Signs Remote Monitoring system.

Today's healthcare units are equipped with the most recent both wired and wireless communication technologies thus making integration of the remote monitoring systems much easier. In the field of Telehomecare, modern internet technologies play an important role, allowing

higher speeds and wider coverage thus turning Telehomecare systems more economically viable. Moreover, the fast development of microelectronic technological devices throughout faster microprocessors, smaller integration modules and an overall less power consuming profile allows systems' developers to produce smaller, more featured and more efficient devices. In the field of vital signs acquisition, smaller sensors are available thus providing more comfort to the patient and more reliable vital signs acquisition.

1.2. Objectives

The aim of this project is to develop a system's prototype whose function is to provide vital signs Remote Monitoring in both hospital and Telehomecare environment. The system must be able to acquire, process and transmit several vital signs' data remotely in order to allow access by a clinician in real-time and anywhere.

Vital signs acquisition must be made using wireless and wearable sensors with low range transmitting capabilities thus allowing vital signs' transmission to a nearby Transmission Module. In order to transmit collected data remotely into a Datacenter, both wired and wireless wide range communications can be used, according to the integration scenario. These include Ethernet, Wi-Fi and GSM/GPRS. Once in the Datacenter, vital signs' data can be accessed via a web-based server or a nursing central.

Besides its communication capabilities, Transmission Module must allow temporary data storage, real-time clock and a certain degree of portability.

In addition to system's development, Knowledge Acquisition is considered as a goal of the project. The acquisition of all the knowledge required to carry out system development's tasks is the primordial goal to achieve. Moreover, during the development process, knowledge should be improved thus resulting in better project's experience.

The presented objectives are mostly related to Transmission Module's firmware programming and the achievement of both sensors and Datacenter integration.

The current project is the continuation of the work performed in the previous academic year by Biomedical Engineering students and therefore the progress made by these students must be taken into account. It is very important to analyse the project's status and to examine the several tasks carried out along the previous academic year in order to avoid redundant work and to detect any flaw that may have occurred.

The Project Reports written by the previous project students can be a valuable help to perform the referred study.

1.3. Document Structure

The intent of the current chapter is to introduce the reader on the project's motivation and the objectives initially proposed.

The following chapter – Project Management – provides information on project's organization such as project members and their tasks division and supervising. A brief overview on the project's planning is also provided.

Third chapter reports most of the knowledge acquired during project's execution. This chapter provides the reader an important background in order to understand most of the thesis' technical concepts.

On the fourth chapter will be discussed the project requirements, more specifically, the Vital Signs acquisition, wide and low range communication technologies and Transmission Module capabilities.

The fifth chapter has the purpose of describing the developed system's architecture. Details on the sensor, Transmission Module and its sockets and Datacenter will be provided throughout this chapter which constitutes important background information in order to reach most of the Firmware Design's considerations.

Chapter 6 – Firmware Design – is the highlight of this thesis. This chapter is divided into each of the Transmission Module's individual part. A detailed description of the most important firmware routines developed for each part is provided throughout this chapter. Communication protocols that are used to communicate among the several system components are also described in this chapter.

In the chapter 7 the results obtained while making several tests after system's conclusion are presented. Furthermore, the conditions in which the tests were made are described.

The last chapter concludes the present document and refers to both the project's status and future work. A student's personal appreciation is also stated with the intent of final conclusion.

CHAPTER 2

2. Project Management

2.1. Project Members

The project team was constituted by three Biomedical Engineering students from the Faculty of Sciences and Technology - University of Coimbra and their supervisors. The development of the Vital Signs monitoring system was carried out by the project students while the supervisors were responsible for the coordination of the project and orientation of the student's work.

The Project Members are presented in the following table.

Table 2.1: Project Members.

Name	Role	Contact
Nuno Varelas	Student	nunomsv@gmail.com
Rafael Simões	Student	rrsimoes.ebm@gmail.com
Tomé Matos	Student	tomermatos@gmail.com
Engineer Paulo Santos	Supervisor	psantos@isa.pt
Professor José Basílio Simões	Supervisor	jbasilio@lei.fis.uc.pt
Professor Carlos Correia	Supervisor	correia@lei.fis.uc.pt
Professor Jorge Landeck	Supervisor	jlandeck@lei.fis.uc.pt
Engineer Lara Osório	Supervisor	losorio@isa.pt

2.2. Tasks Division

The assignment of the tasks that each of the project students would perform was made in the beginning. When applying for the projects, students had to choose the module they wanted to work on.

Remote Vital Signs Monitoring project is divided from the beginning into three modules: Instrumentation Module, Data Processing and Transmission Module and Remote Management Module. The following table specifies the students which each module is assigned to and their respective tasks.

Table 2.2: Tasks Assignment.

Student	Project's Module	Assigned Tasks
Nuno Varelas	Data Processing and Transmission Module	Firmware design of the Transmission Module. Integrate with sensors and Datacenter.
Rafael Simões	Remote Management Module	Datacenter software development.
Tomé Matos	Instrumentation Module	Hardware design.

The presented tasks' assignment was kept until project's conclusion.

2.3. Project Supervising

2.3.1. **Supervising at ISA**

One of the involved entities on the Vital Signs Remote Monitoring project was ISA – Intelligent Sensing Anywhere, which is a Spinoff company of the Department of Physics – University of Coimbra.

The company has a Research and Development department that works on providing complete solutions in several fields like telemetry, industrial automation, environment and healthcare. ISA's R&D department provided an important support on the technical aspects of the project.

Supervision at ISA was mostly made by Engineer Paulo Santos who provided guidance and technical support. Several meetings were made in order to define the project's course and to dissipate some doubts that emerged during work's execution.

The development stage of the project was carried out at ISA's installations, where some of the collaborators provided an important technical support.

2.3.2. **Supervising at CEI**

CEI, the Electronics Instrumentation Centre of the University of Coimbra, is a research centre of the Department of Physics. Its research areas include Atomic and Nuclear Instrumentation, Biomedical Instrumentation, Plasma Physics Instrumentation, Microelectronics, Optical Signal Processing and Telemetry and Industrial Control.

The knowledge acquisition stage of the project was made in CEI's installations which comprises almost half of the project's execution time.

2.4. Project Planning

As it was previously said, this project continues the work carried out in the previous academic year. Therefore, the following tasks were already concluded:

- Hardware design of both Transmission and Acquisition Modules.
- Identification of the features to implement.
- Communication Protocols' definition.
- Flowchart representation of the control processes of both the Transmission and Acquisition Modules.

Face to the encountered project status, the initial planning was to build the first prototype based on the previously designed hardware and start to develop both system's firmware and software parts. This first planning was abandoned because firstly it required wire connected sensors which reduce patients' mobility and secondly, the incorporation of in-house made sensors required time-consuming and expensive medical and electric certifications. After some meetings, the new orientation was to integrate existent wireless sensors thus removing the acquisition module from the system's architecture. The existent Transmission Module hardware was also abandoned and replaced by other one that was already designed and belonged to one of ISA's projects. The new hardware provided the same features as the previous one plus increased modularity. Additionally, it allowed the combination of efforts into just one hardware module that incorporates two different projects.

Later on the project's development, a personalized vital signs remote monitoring solution to be integrated in hospital environment was requested by *Centro Cirúrgico de Coimbra*. The new solution supported SpO_2 and Heart Rate Vital Signs measurement and used Ethernet as wide

range communication technology. The work of the project was then aimed at this new Vital Signs monitoring solution.

Throughout the project's course, ISA asked the student members of Remote Vital Signs Monitoring project to integrate a new project in the healthcare field entitled Look4MyHealth. Detailed description of this project can be found in the project specification (Annex C). The tasks carried out in this project's context consisted on component research, system's architecture definition and elaboration of a project specification document. The amount of time spent with this new project reaches approximately ten percent of the entire project's total time.

CHAPTER 3

3. Knowledge Acquisition

Without a very strong component of this chapter on the project, it would not be possible to accomplish it, which turns knowledge acquisition into one of the most important and most necessary tasks carried out along this year. The existent knowledge on the field of microelectronics and firmware programming was practically inexistent, mainly due to the lack of this kind of subjects during Biomedical Engineering's course. This limitation was overcome with a big amount of curiosity and interest for this matter allied to the effort applied on this task. The help supplied by some ISA's engineers was also a good contribute to overcome the previously mentioned limitation.

The acquisition of new concepts was the first phase of the project to be started and the last one to be accomplished. A variety of concepts were approached along the entire year in a relatively wide range of subjects. These include, mostly, C programming, microcontrollers, development environment, simulation software and Bluetooth communication.

The following topics try to give a brief description of the mentioned concepts and technology in order, not only to allow a better understanding of the more technical thesis's chapters, but also to give a good perspective of the work carried out along the course of the project.

3.1. C Programming Language

C language was originally developed at Bell Labs between 1969 and 1973 by Dennis Ritchie and Brian Kernighan. The development of the Unix operating system, at first implemented in Assembly language on a PDP-7, promoted the origin of C whose first implementation platform was a PDP-11 running under Unix. Since then, C language has evolved and is now standardized in the Computing Industry as a well-known system implementation language (5).

C's most particular features are its absence of restrictions and the reduced number of language components. C language is terse, that is, if two language features accomplish almost the same thing, only one is included. In spite of not being a "very high level" language nor a "big one", these particular features, make C a more convenient and effective language for many tasks than supposedly more powerful languages (5).

On the other hand, the same particular features can become an obstacle for an inexperienced and beginner programmer. Error checks exist only at compile-time but sometimes even after the compiling process, there is no assurance that the code will perform as planned. The code runs in a minimal run-time model that has no safety checks for bad type casts, bad array indices, or bad pointers. Also, there is no garbage collector to manage memory. Instead, the programmer manages heap memory manually. These particularities make C a fast, objective but fragile programming language (5).

3.1.1. **Structure of C Programs**

All C programs follow a predetermined structure, constituted by several containers that make the programming process easier and structured. Each part of this structure is responsible for a certain type of information (6).

Pre-processor Directives

The Pre-processor Directives are constituted by commands that are interpreted by the C Pre-Processor. The Pre-Processor is a separate program, invoked by the compiler at the beginning of a compiling process, so that the information contained by these directives can be taken into account in the compiling process. The two most common Pre-Processor Directives are *#define* and *#include*. The first substitutes a text for a specified identifier while the second includes the text of an external file into the program.

Declaration

In order to define the name and type of variables and functions in a C program, a declaration is necessary, since the declaration of a variable or function depends on its use. In a declaration, several attributes can be defined:

- Basic Type: *char*, *int*, *float* and *double*.
- Signedness: *unsigned*, *signed*.
- Size: *short*, *long*.

– Type Qualifiers: *const* (variable cannot be modified).
– Pointer Types: by means of the * type declarator.
– Arrays: by means of the *[ ]* type declarator, following variable name.

Definition

In a definition, the contents of a variable or function are established. The type of contents that can be established during a definition depends on the declaration. In a definition it is also allocated the storage needed for a given variable or function.

Expression

A C program is mostly constituted by expressions. An expression is every piece of code that combines operators and operands in order to obtain a single value.

Statements

Statements are responsible for the flow control and order of a program execution. These are constituted by reserved C words that must be used with appropriate syntax. It is necessary to know how to use statements in the proper way in order to achieve the program's objective.

3.1.2. **Why use C in Microcontroller Programming**

Generally, the typical choice for embedded systems programmers' has been, for many years, the Assembly language. Nowadays, face to the offers of the many microcontrollers' suppliers, this tendency has been fading out. The current microcontrollers can, not only can be programmed using high-level C programming language as well as Assembly, but also were specially designed to use C language as means of programming. This last fact eliminates some of the obstacles introduced by high-level languages in microcontrollers' programming (7).

The three major benefits of a high-level language are portability, readability and modularity. The use of a high-level language allows the programmer to discard the majority of the specificities of the target hardware and focus on a functional implementation. This enables the possibility for the programming source to be re-programmed to a different hardware, only with a few changes, thus conferring portability to the source code. A high-level language allows for much easier reading and writing due to the similarities with the human's language. This increased readability makes programs easier to build and maintain. Modularity, on the other hand, allows the programmer to divide an entire source code into functions, each one with a specific purpose.

These modules can be included then into various projects or used several times in the same project. A decrease in a project's development time can be achieved applying the modularity programming concept (7).

Despite all the advantages of using C language for microcontroller programming there is, however, a consequence regarding the compiler's effectiveness. Code efficiency is often reduced when programming in a high-level language as the compiler may not translate the C source code to the optimum machine code. To overcome or minimize this issue the programmer should select a microcontroller designed to be programmed with C language and study the compiler in order to determine the best way to write code efficiently (7).

3.2. Microcontrollers

The essence of this project lies in microcontroller devices. During almost the entire project development time there was a close contact with this kind of devices. Due to its easiness of programming, debugging capability, extensive documentation support and the fact of being widely used in the field of electronics, the PIC microcontroller unit was chosen to be the object of study and thus the object of work. Among the several stages of the learning and developing process, several families of the PIC microcontrollers were used. The initial contact was made with an 8 bit, 18F family PIC mounted into a demonstration board (PICDEM Z) from Microchip.

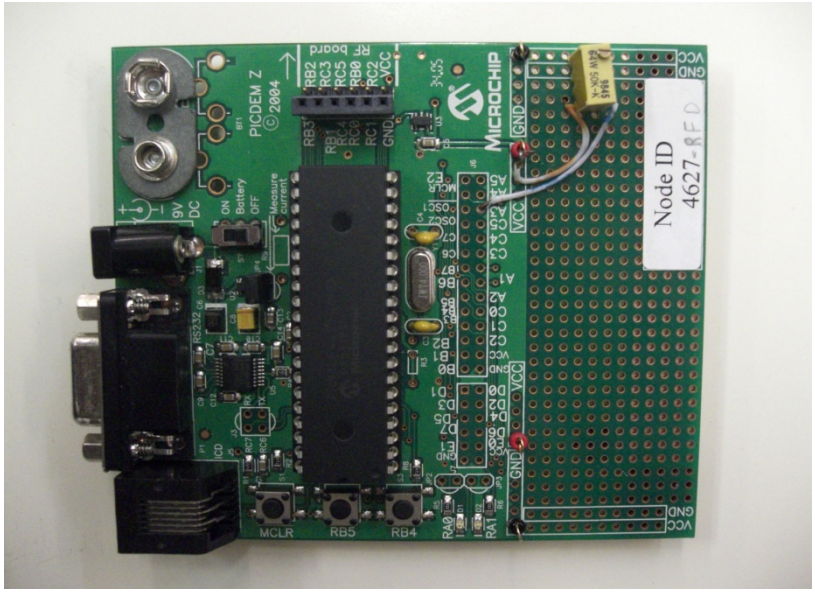


Figure 3.1: PICDEM Z demonstration board with PIC18F4620. This board was used to learn the first firmware programming concepts.

Due to project's development requirements a learning of the 16 bit architecture PIC had to be made. In order to accomplish this stage a virtual demonstration board (Explorer 16) from Microchip running in Proteus VSM was used (see chapter 3.4). Besides the 24FJ128GA010 PIC, this demonstration board comprises several peripherals such as I_2C and SPI thermometers and memories that allowed the elaboration of firmware code to control this kind of devices.

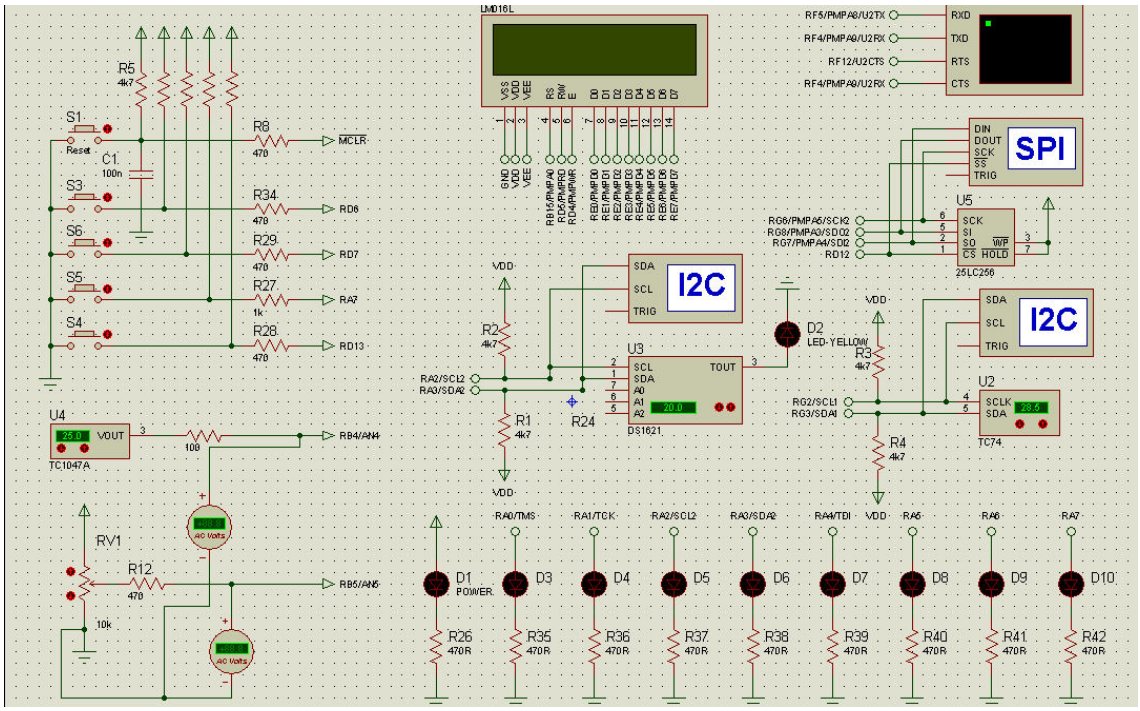


Figure 3.2: Explorer 16 Virtual demonstration board with PIC24FJ128GA010. This virtual scheme was used for 16 bit PIC learning.

A microcontroller is called a computer-on-chip due to its high integration. This means that in the same chip, besides the microprocessor, are contained data and program memories, input/output interfaces and peripheral modules. This high integration makes these devices capable of storing and running a program by themselves. Microcontrollers operate at low clock speeds, face to the recent microprocessors. This particularity makes them suitable to typical applications that require low power consumption but low processing capabilities (8).

The following topics will fall upon the PIC microcontroller which was, as said, the object of study and work during most of the project's time. The PIC microcontroller is based on a modified Harvard RISC instruction set architecture. This kind of architecture has the advantage of having few internal instructions which simplifies the learning process (9; 10).

3.2.1. **Central Processing Unit**

The main part of a microcontroller is the Central Processing Unit (CPU). This part is responsible for finding and fetching the instructions that need to be executed so that they can be decoded and finally executed. The instructions written by the programmer are initially stored in the Program memory. The CPU has the job of fetching these instructions and decoding them into a set of actions that perform an assigned task. These actions can comprise transfer of data between memories, from memory into ports or the performing of operations; and for that reason the CPU must be connected to all parts of the microcontroller (10).

3.2.2. **Memory**

The function of this part of the microcontroller is to store data. This kind of devices operates according to the addressing concept. According to this concept, the memory is divided into several blocks, each one with a specified address. This way, when manipulating data, the address must be provided in order to assure the access in the right location.

The PIC microcontroller memory is organized into two separate memory blocks; one for data and the other for program. Generally, the program memory stores the list of instructions and is carried out in Flash technology thus allowing the programming of a microcontroller several times. Data memory, on the other hand, generally consists of EEPROM and RAM memories. EEPROM contents are not lost during power supply loss which makes it suitable for permanent values storage and can be accessed during program execution. RAM memory, on the other hand, contains the microcontroller registers (10).

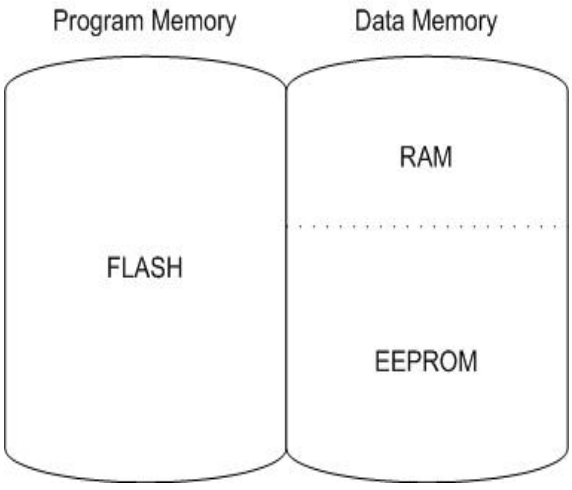


Figure 3.3: PIC Microcontroller's most common memory organization.

3.2.3. **I/O Ports**

In order to allow the connection between the microcontroller and peripherals, a number of input or output pins must be used. These pins are grouped into I/O ports on which a desired value can be set or an existing one can be read. An I/O port can be a group of 8 or 16 I/O pins and each one is mapped into a register inside the microcontroller. This way, the I/O port registers inside the microcontroller are physically connected to the pins that allow the connection to the outside peripherals. Besides the basic digital port function, some of the I/O ports' pins are multiplexed with other microcontroller internal functions like timers, serial communication, oscillator, etc. It is part of the programmer's job to assign the right function to the I/O ports' pins as well as to set their direction (input or output) (11).

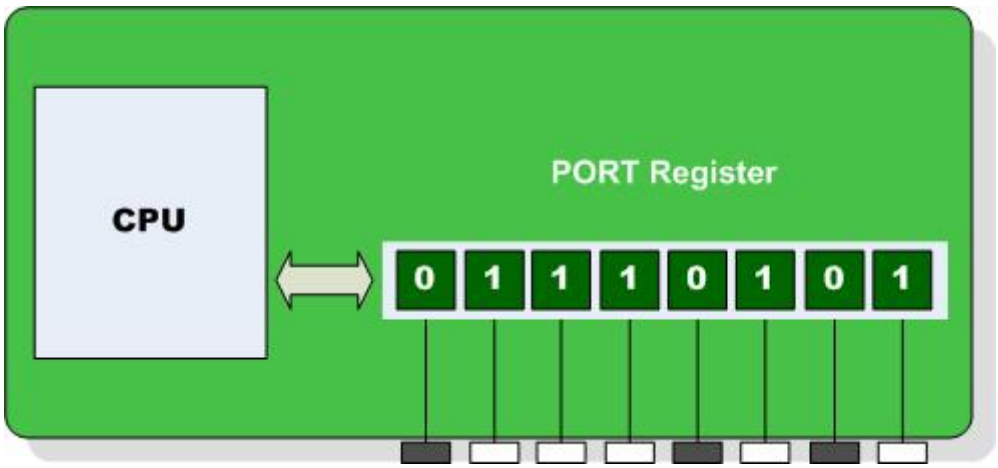


Figure 3.4: Illustrative scheme of an 8 bit I/O port and the respective pins and register couplings. Note for the pins' state, consistent with the port register values (grey for 0 and white for 1). For simplification purposes just one I/O port is represented.

3.2.4. **Timers**

In order to elaborate a complete and functional system's firmware it is almost indispensable to introduce the notion of time into the program. This is done using the microcontroller timer units. These units are based in a counter register whose value is incremented every time a given pulse occurs. Most of the times, the referred pulse is supplied by a calibrated oscillator with a known and precise frequency. This way, by taking the counter value in two different time instants, it is possible to establish a relation between time dimension and the counter variable. The incrementing process is made automatically and in the background which

makes the programmer's task easier. Timer units have a wide range of practical applications and it is the programmer's job to find out the best way to take advantages of all the features (10; 11).

3.2.5. **Serial Communication**

The need to communicate with larger amounts of data promoted the Serial Communication. This way, it consists on sending a sequence of data bits along the same line with a predetermined frequency. Two devices serially connected with each other can, this way, communicate unlimited amount of data with a reduced number of lines between them. However, in order to serial communication to work, the rules of exchange of data must be set in advance. These rules are called the serial communication protocol.

In order to make integration easier, most PIC microcontrollers have peripheral modules that already implement some of the serial protocols. Generally these modules include USART, SPI and I^2C communications. Next, it will be given a brief overview on these communication modules (10).

USART

USART stands for Universal Synchronous Asynchronous Receiver Transmitter. As the name points out, this module implements both synchronous and asynchronous communication. In order to establish a connection between two devices only two lines are needed (11).

In the asynchronous communication the lines are called TX and RX, for transmission and reception respectively. This means that it is possible to operate in full duplex or, in other words, both transmission and reception can occur at the same time, however the transfer rate must be previously set. The most common use of USART module is in asynchronous mode, in order to implement the so called RS-232 standard (11).

The synchronous mode, on the other hand, uses a data and clock line. The use of a clock line allows communication without a previously fixed transfer rate however the existence of only one data line limits the operation into half duplex. The synchronous communication mode operates in a Master-Slave philosophy in which the Master device always provides the clock line.

Features like 9-data bits format, *buffer* overrun detection and parity and stop bits configuration are also supported by the USART Module (9).

The connection lines exchanged between two devices in both asynchronous and synchronous serial communication are shown in Figure 3.5.

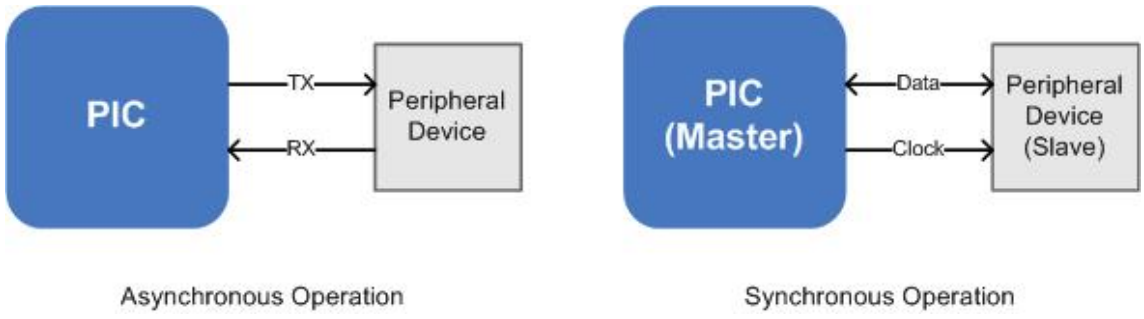


Figure 3.5: Illustrative scheme of the connection lines exchanged between two devices in both Synchronous and Asynchronous Serial Communication.

SPI

SPI, which stands for Serial Peripheral Interface, is a synchronous protocol that allows a Master device to exchange data with several Slave devices. Generally, this protocol allows high-speed communication allied to an easy implementation. Since SPI is a synchronous Master-Slave protocol a clock line (SCK), controlled by the Master, is required to connect the devices. Regarding data transmission SPI is a data exchange protocol, in other words, as data are being clocked out, new data are simultaneously being clocked in. The data transfer lines are called SDO and SDI, for output and input respectively. As said, SPI protocol supports several Slave devices connected to one Master device. This multi-Slave operation can only be achieved using a Slave Select or Chip Select line, thus allowing the Master device to signal the Slave device he wants to communicate. Exchanged connection lines in SPI protocol are shown in Figure 3.6 (11).

PIC microcontroller SPI module automatically handles the lines involved in SPI communication which makes the programmer's job much simpler (9).

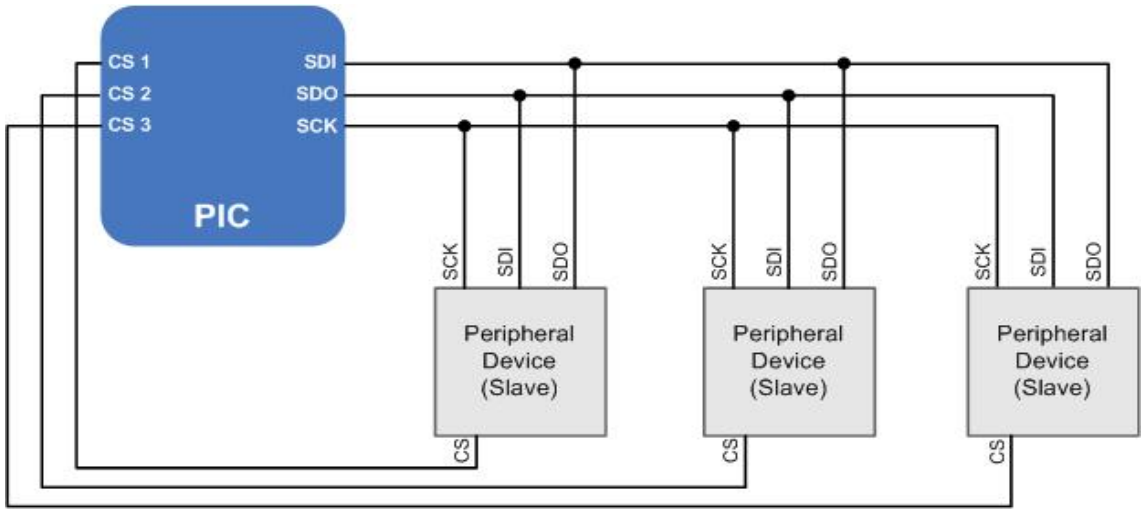


Figure 3.6: Illustrative scheme of the connection lines exchanged between microcontroller (Master) and several peripheral devices (Slaves) in SPI communication

I²C

The I_2C (Inter-Integrated-Circuit) module of the PIC microcontrollers allows communication between two or more devices at a high-speed and is more commonly used to communicate with other microcontrollers or smart peripheral components like memories, real-time-clocks, etc.

Similar to the SPI, I²C protocol is synchronous and implements a Master-Slave communication. On the other hand, only two lines are needed to exchange data between devices. Since the protocol is synchronous, one Master controlled clock line is always required and is called SCL. In order to transport the data bits, only one bidirectional data line is implemented (SDA). With the intention of minimizing the data collisions in the bidirectional data line, an "Acknowledge" (ACK) system is also implemented in the I_2C protocol. The receiver must issue an acknowledge condition thus informing the sender that the data was successfully received. Unlike in SPI protocol, which has a dedicate line to perform device addressing, in I_2C mode the addressing procedure is made through data exchanging. Each Slave device has a 7-bit address so that when the Master wants to initiate the communication process, it must send into the data line a first byte with the address whose Slave device it wants to communicate. From that point on, all the sent data will be processed by the addressed device and ignored by the remaining ones. With the intent of improving even more the communication reliability Start and Stop conditions are also implemented indicating, respectively, the beginning and end of an I_2C communication process. (9; 10; 11)

Figure 3.7 presents the exchanged lines between several devices in I_2C serial protocol.

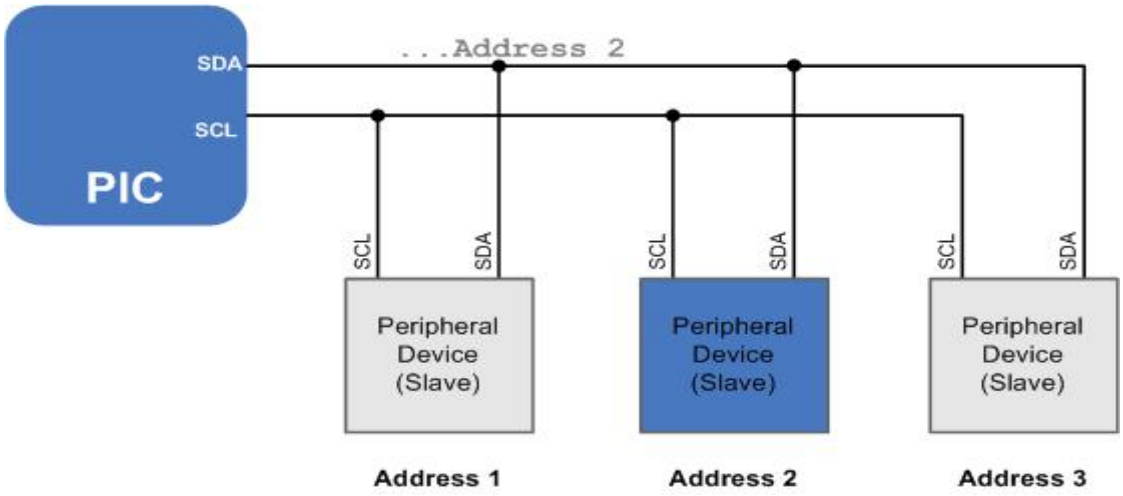


Figure 3.7: Illustrative scheme of the connection lines exchanged between microcontroller (Master) and several Peripheral devices (Slaves) in I_2C Protocol. Note for the address sequence sent along the SDA line towards the second Slave Device (in blue).

Like in the other serial communication modules, PIC microcontroller's I_2C module automatically implements all the timings and bus transitions of the protocol thus minimizing programmer's margin of error (9).

3.2.6. **Analog Module**

Most PIC microcontrollers are prepared to recognize not only logical zero or one values but also analog precise voltages. This is made, of course, by means of an embedded Analog to Digital Converter. Despite there is only one ADC, generally it is internally multiplexed over several analog input pins. The PIC microcontroller ADC module generally generates a 10-bit binary result using the method of successive approximation and is able to store it on the PIC ADC's internal registers. This way, depending on the acquisition time, it is possible to sample a given signal with a predetermined sampling rate. The analog module allows, among other options, the selection of the acquisition time and upper and lower voltage references in order to adjust ADC's conversion both accuracy and precision as well as voltage resolution, respectively (10; 11).

3.2.7. **Interrupt System**

Sometimes, it is necessary to handle events instantly while in the normal program execution. The Interrupt System is a mechanism of the microcontroller that allows it to stop the normal program execution, execute a subroutine that handles the interrupt and afterwards resume the normal program execution from the exact point it stopped (see Figure 3.8).

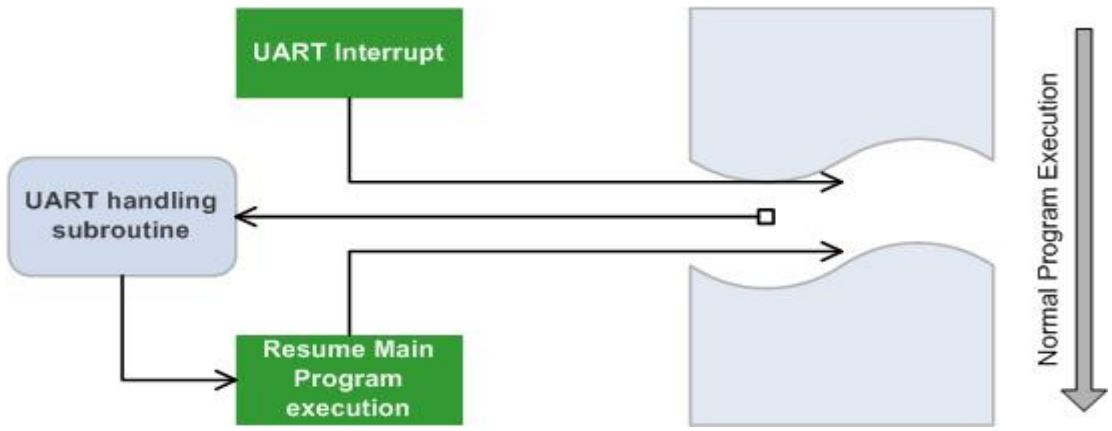


Figure 3.8: Illustrative scheme of an Interrupt procedure. The UART interrupt source is just an example as the presented procedure applies to all interrupt sources.

Interrupts are often used to handle communications and extern triggers like pushbuttons however they can be used together with timers in order to enable periodic routines like a led tick, device pooling, local variables refreshing, etc.

In a multifunctional system, interrupts are one the most important features of the microcontroller, therefore the programmer must dedicate many of his efforts in the correct configuration of the Interrupt System (11).

3.3. Development Environment

Just like a biologist needs to set up his lab in order to develop a new vaccine or a writer to set up his desk in order to develop a new story, a firmware programmer needs to set up and define his Development Environment before starting to develop code.

The process carried on in an application development, also called Development Cycle, is constituted by four main steps that form a cyclical operation. These include Edit/Create/Design code, Compile/Assemble/Link code, Program device and Analyze/Debug firmware (see Figure 3.9). As all these steps, from design to implementation, cannot be done flawlessly at the first time, they all link together forming a cycle (12).

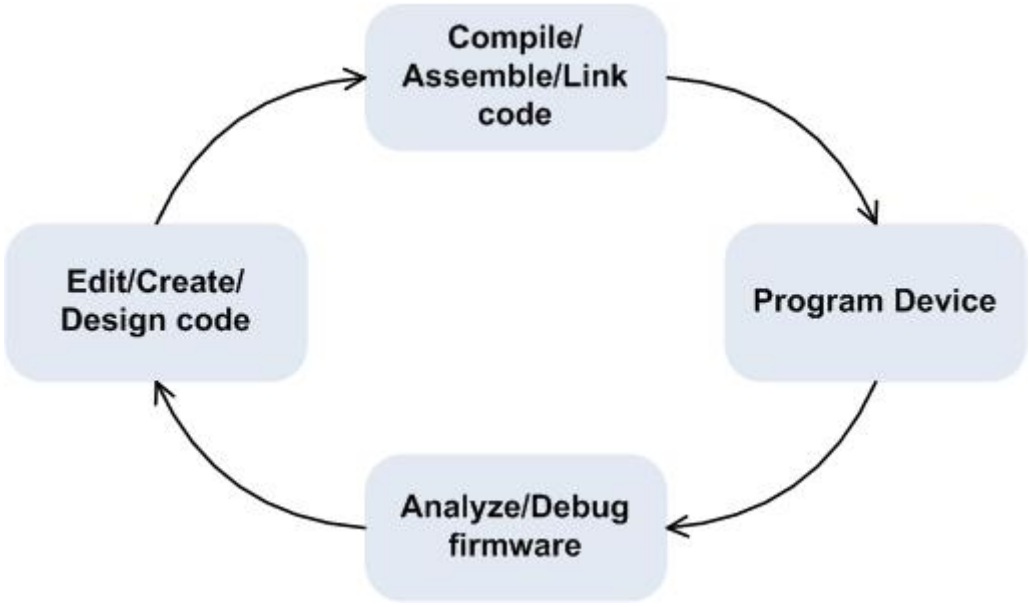


Figure 3.9: Development Cycle.

Once aware of the Development Process' flow it is possible to identify all the Development Environment's components. These include a Compiler, a Programmer/Debugger

and an Integrated Development Environment software. The following topics will not only give a brief overview on each of the components but also present the reasons that promoted their choice.

3.3.1. **Compiler**

As it was previously mentioned, during both the leaning and development process, two different PIC microcontroller architectures were approached. Since the compiler is directly related to the hardware architecture, two different compilers had to be used. HI-TECH PICC-18™ was used to handle 8-bit PIC microcontrollers while MPLAB C30 handled the 16-bit devices.

A compiler is a piece of software that applies a number of relatively simple rules in order to translate high-level language statements like C statements into Assembly language. Modern compilers have advanced techniques in order to simplify and optimize assembler code, thus turning the programmer's task the more transparent as possible. In fact, the correct term for this kind of compilers is cross-compiler, since it is running on a platform other than the one whose code is being built (12).

While selecting the compiler, the first and most important aspect taken into account was the full-feature compliance with the ANSI C language. This feature is very important in order to minimize the user integration impact into different compilers. Once the programmer knows how to manage standard C language, few differences will be noticed when using a full-featured ANSI compliant C compiler. Both HI-TECH PICC-18™ and MPLAB C30 are ANSI C - full featured compilers (13; 14).

To a programmer, one of the most important support tools is the available documentation concerning the subject of work. Generally, the availability of books, articles, tutorials, problem solving forums and sample code is directly related to how many people uses a particular compiler. Both HI-TECH and C30 are widely used PIC microcontroller C compilers which results in a large documentation support whether by the official manufacturer or other programmers in an individual or academic context.

Other feature that was taken into account was the ability to make the programmer's task easier, in other words, the amount of built-in functions and macros needed by the programmer in order to control peripherals or access microcontroller's registers. Both HI-TECH and C30 compilers have not only these low-level controlling macros and functions but also memory management, input/output and other general purpose C libraries thus saving some tedious work to the developer (13; 14).

Finally, an easy Integration with an Integrated Development Environment was also an important aspect taken into account. Integrated Development Environment software introduces a very significant reduction in the time spent on all the four Developing Cycle stages (see chapter 3.3.3). MPLAB C30 compiler was planned with MPLAB IDE integration in sight, while for HI-TECH it exists a tool suite software that allows an easy integration with the same IDE software.

3.3.2. **Programmer/Debugger**

Once compiled, assembled and linked, the program must enter the microcontroller's programming memory. The process of transferring the final firmware into the chip is called Programming and after this procedure, the microcontroller is running the program by himself.

One of the particularities of microcontrollers' programming is that when something goes wrong there is no output so that the developer can find the problem. This enables the use of an In-Circuit-Debugger, which allows the programmer to debug through the code at the same time it is running on the PIC, making the developing task much easier. This device consists on a hardware piece that stands between the developer's PC and the system to be tested, connected by wires into these two units.

During the developing process, an In-Circuit Debugger and In-Circuit Serial programmer device was used in order to achieve maximum developing efficiency. The used device was MPLAB ICD2 from Microchip, the same manufacturer of PIC microcontrollers. Real time and single-step code execution, breakpoint setting and registers and variables watch or modify are the major aids that ICD2 provides to the developer. In addition, it allows the normal programming procedure (15).

Figure 3.10: ICD2 picture. Note for both the PC and microcontroller connections.

3.3.3. **Integrated Development Environment**

The main part of the programmer's Development Environment is a piece of software called Integrated Development Environment or IDE. As the name points out, IDE software provides a single integrated environment with all the facilities that a programmer needs in order to develop and test code. IDE can be compared to a workbench with all the needed tools built-in. The purpose of the IDE is to maximize programmers' productivity by providing an integrated and user-friendly single interface, in other words, programmers can replace tedious command invocations in several applications by button clicking or shortcuts in the same software application.

When one refers to PIC microcontrollers' IDE, MPLAB IDE from Microchip is the first name that comes to mind. Besides it is widely used among PIC programmers, it is maybe the only option that offers total integration with both compiler and debugger/programmer.

Next, it will be given an overview on the most used MPLAB's built-in components. These include the Project Manager, Editor, Assembler/Linker and Debugging Tools.

Project Manager

Project Manager not only helps to organize the different source files that compose a firmware project, but also establishes the connection between IDE and the Language Tools (12).

Editor

The editor is an indispensable component in the IDE software, since it not only performs as a source code text editor but also constitutes a support to the debugger module. MPLAB's text editor is poor-featured compared to other single-purpose source code text editors however C language does not require powerful text-editing features due to its simplicity (12).

Assembler/Linker

Assembler is responsible for taking the Assembly code returned by the C compiler and turning it into object code so that it can be programmed into the microcontroller. When a project comprises several source files, the assembler must be used with the Linker. The Linker takes several objects generated by the assembler and links them into a single executable program. The linker also positions the program code into the correct memory areas of the microcontroller (12).

Debugging Tools

MPLAB's debugging tools comprise almost all the features of a normal software debugger. These include breakpoints, single stepping and watch/modify variables and were used with MPLAB ICD2 device (see chapter 3.3.2). The debugging feature has its limitations; the synchronization between the IDE software and microcontroller is slow and sometimes fails, however the existent method is a precious help to PIC programmers (12).

3.4. Firmware Simulation

Firmware simulation consists on testing a microcontroller program in a simulated software environment, thus replacing physical hardware into a computer software program. In the project's context, the employ of a firmware simulation tool constituted a big help in both the firmware programming learning and development processes. Proteus VSM from Labcenter Electronics was the software used to accomplish firmware simulation. It is maybe the only existent software program that allows both PIC microcontroller's simulation and MPLAB IDE integration.

3.4.1. **Proteus VSM**

Proteus VSM allows co-simulation of microcontroller based designs by means of a combination of SPICE circuit simulation, animated devices and microcontroller and other smart integrated circuits simulation engines. The developer can interact in real time (sometimes only near it) with the design as if it were a real prototype, using LED indicators, displays, communication debuggers, etc.

The main advantage of using this kind of software is that it allows building a simulated hardware prototype in record time, compared to the traditional hardware development process. In the project's context, Proteus VSM allowed the access to a full-featured PIC microcontroller demo board in order to test all the firmware routines.

Capabilities

Firstly, Proteus VSM has an easy-to-use schematic design tool and library that allow the developer to design the hardware he needs. The analogical component of the design is simulated by a dedicated analog simulation engine which includes a large number of device models, most

of them supplied by the respective manufacturer. In order to perform an accurate hardware debugging, there are several virtual instruments included. These include oscilloscope, logic analyzer, virtual terminal and SPI and I_2C protocol analyzers. The last tools provide an important help to test communication routines between digital devices. Finally, the most important and interesting feature of Proteus is the ability to program or debug virtual microcontrollers with real firmware and test its interaction with other digital or analog devices connected to it. Allied to the referred features there is a diagnostic messaging system that not only specifies the events that occur in the whole design but also warns the developer about the errors that take place and the source of the problem which is a valuable help to every firmware programmer (16).

Limitations

During the period that Proteus VSM software was used, some limitations were encountered, in spite of the global positive note.

The first found limitation was the lack of the most recent PIC microcontroller virtual models available for simulation however the same did not happen for the most used peripheral devices. Secondly, inconsistencies in some peripheral's virtual models were found which conditioned the developing process, especially in SPI and I_2C enabled peripheral devices. Finally, some real-time simulation issues were found. Sometimes the virtual engine could not maintain real-time synchronization between microcontroller's code execution and virtual hardware events. This limitation was often detected when working with Timers, Real-Time clock and delay routines.

MPLAB IDE Plug-in

Integration between VSM and MPLAB IDE is provided by Proteus VSM MPLAB IDE plug-in which allows the aggregation in the same software program, of both the development environment and the virtual hardware. In other words, when the programmer wants to test his firmware program, he just needs to start simulation to see things happen right in the MPLAB's window. Compared to the standard process, Proteus VSM and MPLAB IDE plug-in allow a considerable time reduction in firmware developing (16).

3.5. Bluetooth Communication

The wireless short-range communication protocol chosen to collect vital signs data was Bluetooth (see chapter 4.2.2) and, therefore, some knowledge had to be acquired in order to

understand the architecture, operation procedures and limitations of the referred protocol. Furthermore, a deep study of the hardware used to establish the Bluetooth connections had to be made in order to understand their operation and successfully integrate them into the system. The referred hardware consisted on a Bluetooth Module and Bluetooth oximeter. The hardware selection process was practically inexistent. At ISA it already existed these two hardware devices and due to time constraints and the high-cost associated to this kind of devices, the choice was to use the existent hardware, at least in the first stage of the prototype. The Bluetooth module used was Free2move's F2M03GLA while the Bluetooth oximeter was Nonin's 4100 Bluetooth oximeter. In the following topics, it will be given a brief overview on the Bluetooth protocol as well as both the Bluetooth module and oximeter devices.

3.5.1. **Protocol Overview**

Bluetooth is one of the most popular short-range wireless protocols among electronic devices. Bluetooth technology consists of radio frequency technology and protocol software. This enables Bluetooth capable devices to communicate in short-range and off the line of sight, with a predefined amount of rules. Both voice and data can be transferred between Bluetooth capable devices (17).

Operation

Bluetooth's Radio Frequency operates in the unlicensed Industrial, Scientific and Medical band at 2.4 GHz. Its operation is based on a spread spectrum transmission with frequency hopping which means that it allows several channel communication and uses that feature in order to minimize wave interferences. In other words, when a device wants to transmit according to the Bluetooth standard, it must check for the presence of other devices in the same frequency and therefore try to avoid the frequencies in use. The mode of communication for Bluetooth technology is based on the synchronization between one Master device and all the other devices, known as Slaves that are connected in the same network. This group of Bluetooth enabled devices that communicate each other is called piconet. The devices of the same piconet communicate via a common physical radio channel whose synchronization clock is provided by the Master device (18).

Besides the radio frequency technology, Bluetooth relies on multi-layer protocol software that allows Bluetooth enabled devices to understand each other. The referred protocol software, also called the Bluetooth Stack, is already integrated into most of the market available Bluetooth

modules thus making protocol issues totally transparent to the developer. In other words, the developer does not need to know the Bluetooth protocol software in order to control Bluetooth connections (18).

Classes and Versions

Bluetooth standard was designed with low power consumption in mind which appears to be one of the advantages of this kind of technology. Power consumption, however, strongly depends on the transmitter's signal power and range. In order to segment Bluetooth devices according to maximum power allowances, several classes have been defined (see **Table 3.1**).

Table 3.1: Bluetooth Protocol Classes (18).

Class	Maximum Permitted Power mW (dBm)	Range (typical values)
Class 1	100 mW (20 dBm)	~100 meters
Class 2	2.5 mW (4 dBm)	~10 meters
Class 3	1 mW (0 dBm)	~1 meter

When choosing a Bluetooth device, the application and the required transmission range must be taken into account in order to select the preferable power class.

Since Bluetooth communication was first introduced, several versions have been released. The first Bluetooth versions were 1.0 and 1.0B which had too many problems and were practically discarded by manufacturers. The first successful Bluetooth version was 1.1, were most of the interoperability problems of the earlier versions were corrected. In spite of many of the Bluetooth enabled devices uses this version, some manufacturers are using versions 1.2 and 2.0 to develop modern devices. Besides the obvious stability improvements, the main difference between versions 1.2 and 2.0 is that data rates vary from 1Mbps in the older version to 3Mpbs. All Bluetooth versions are backward compatible (19).

Pairing

The term "Bluetooth Pairing" is the used term to indicate that two Bluetooth devices agreed to establish a communication with each other. Before the Pairing occurs, a password (passkey) must be shared between each device, otherwise the pairing process will fail. The passkey can be known in advance by the devices or negotiated in the Pairing process. Each of the Paired Bluetooth devices can end the connection at any time (18).

Security

Bluetooth, like all communication technologies, faces the problem of security and the problem emphasizes being Bluetooth a wireless technology. There are, however, some security options provided by Bluetooth protocol that can be implemented by the developer. The first security barrier is the Pairing process. In order to link two devices a passkey is needed (see chapter 3.5.1 - Pairing) which prevents Paring with undesired devices. Once paired, transferred data between Bluetooth devices is encrypted, thus denying data access to undesired entities (18).

3.5.2. **Bluetooth Module**

As it was previously mentioned, in order to enable the Vital Signs' Transmission Module with Bluetooth connectivity, a Bluetooth Module was used (Free2move F2M03GLA). Due to its relative complexity, the module had to be deeply studied before starting to develop microcontroller routines to handle it. In order to understand many of the concepts presented on both the device's data sheet and communication Protocol, some knowledge had to be acquired.

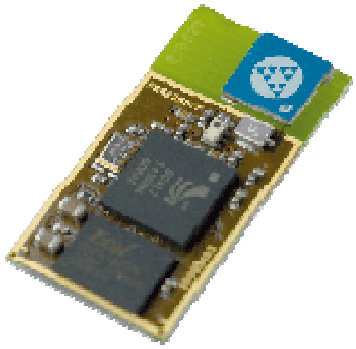


Figure 3.11: Free2Move F2M03GLA Bluetooth Module (20).

The Free2move F2M03GLA is a low-power embedded Bluetooth module with built-in antenna which means that no extern components are needed besides the module and a host processor in order to handle Bluetooth connections. Internally, the module implements Bluetooth version 2.0 and is configurable as Class 1/2/3 (see chapter 3.5.1 - Classes and Versions). The used Bluetooth Module implements an easy-to-use firmware called Wireless UART which implements the Bluetooth Serial Simple Pairing (SSP). The purpose of the Wireless UART firmware is to replace serial cables that connect electronic devices. In other words, it is very easy

to emulate a serial connection between two Bluetooth enabled devices. Once a successful Bluetooth connection is established, data are exchanged transparently between host processor and the remote Bluetooth device. The main disadvantage of using Wireless UART firmware is that only one Bluetooth connection can be established at a time. In a situation of a piconet with more than two devices, Wireless UART becomes ineffective, since it only allows Pairing with one device at a time. This constitutes the main limitation that was found in this module (21; 22).

3.5.3. **Bluetooth Oximeter**

In order to measure SpO_2 and heart rate vital signs Nonin 4100 Pulse Oximeter with Bluetooth was used. So that this device could be integrated into the Vital Signs system, some aspects concerning its Bluetooth operation had to be fully understood.

Figure 3.12: Nonin 4100 Bluetooth Oximeter (23).

Nonin 4100 is a Class 2 Bluetooth device and, according to manufacturer, is specified for a range about 30 feet, thus resulting in a 120 hour battery life. Bluetooth connectivity of the oximeter provides Serial Port Profile (SSP) support and uses Bluetooth Version 1.1. Regarding communication, Nonin 4100 is considered a Slave device inside a piconet. The communication between the oximeter and other devices can be made point-to-point or point-to-multipoint. In point-to-point communication, one Master device is connected only with the oximeter. When connected both devices become invisible to other Bluetooth devices. On the other hand, when in a point-to-multipoint piconet, the Master device can communicate with up to seven Bluetooth enabled devices, including Nonin 4100 Oximeter (24; 25).

Although the Wireless oximeter supports both point-to-point and point-to-multipoint communication modes, the integration with the used Free2move Bluetooth Module implies a point-to-point communication. As seen in chapter 3.5.2, F2M03GLA Module with Wireless UART firmware only allows Pairing with only one device at a time thus discarding the possibility of forming a piconet with more than two devices (see Figure 3.13).

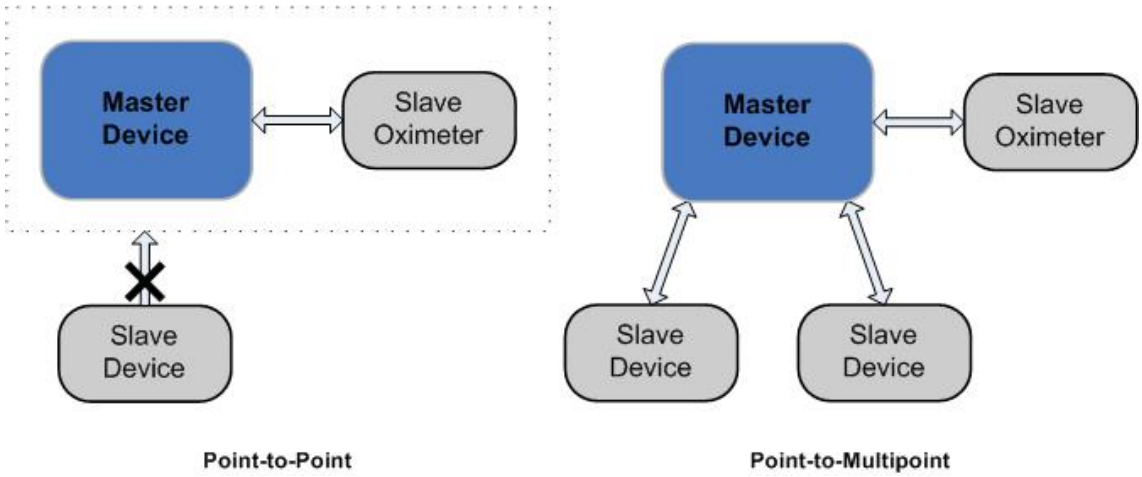


Figure 3.13: Nonin's oximeter piconet organization in both point-to-point and point-to-multipoint communication modes. Note to Master-Slave invisibility face to other Slave devices in point-to-point mode.

Chapter 4

4. Project Requirements

Before taking any action in system's development stage, all system requirements need to be identified. Sometimes, a modification in the system architecture while in the development stage may lead to the loss of already done work or error-introducing adaptations. A system's development process will run smoother if the definition of its requirements is done entirely in advance.

The first step documented in this chapter is related to the presentation of the past year previously selected sensors. Some background information will be provided in order to allow fully understanding of the chosen acquisition sensors. Importance of each sensor will also be discussed and limitations will be found in order to conclude over the sensors' relevance and clinical validity on the system's context.

Moreover, several short-range and wide-range communication technologies will be presented in order to explain the choices made along system architecture definition. The role of each presented communication technology will be indentified regarding the system's integration scenario.

Finally, hardware required capabilities will be identified and discussed in the entire system's context.

4.1. Vital Signs Acquisition

Vital signs acquisition is an important aspect on the Vital Signs Monitoring System. The type of sensors used in the system must be analyzed in order to conclude over the validity and importance of the vital signs that each sensor measures. The purpose of the vital signs' system is to allow clinicians to infer over human beings' physical conditions, so the measured vital signs must be related as much as possible with the occurrence of health conditions.

From the last year's project there were four sensors already chosen which include oximeter, electrocardiograph, accelerometer and thermometer. This section's purpose is to analyze the sensors that were chosen to integrate Remote Vital Signs Monitoring system.

4.1.1. **Pulse Oximeter**

Pulse oximeters provide an indirect and non-invasive measure of oxygen saturation of haemoglobin in the arterial blood as well as changes in the blood's volume which allows the indication of the heart rate value.

Oxygen saturation is based on the principle which states that the absorption of light at two different wavelengths differs according to the haemoglobin's degree of oxygenation. Oxygen saturation is then determined by measuring light absorption at the wavelengths of 600-750 (red) and 850-1000 (infrared) nanometers. The ratio of the proportions of the absorbed light at each frequency is then calculated and compared against direct measurements of arterial blood oxygen saturations. Due to the changing volume of arterial blood with each pulse beat, the light signal collected in the oximeter has pulsatile component. This pulsatile component allows the separation of the absorption due to arterial blood from the absorption due to tissues and venous blood (26; 27).

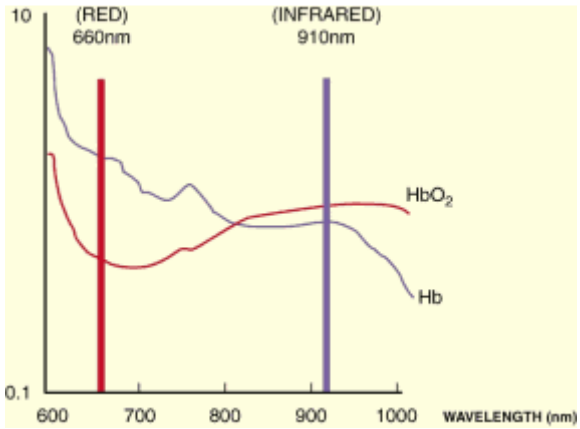


Figure 4.1: Light extinction coefficient for both reduced and oxygenated haemoglobin versus wavelength. Vertical axis is in logarithmic scale. Note for both the red and blue vertical bars which indicate the measured light wavelengths (28).

Pulse oximetry is a widely used technique over various clinical applications because it offers the monitoring of the cardio respiratory status of patients. The most common ones are anaesthesia, recovery, intensive care and patient transport (26).

The technique has, however, some limitations which may affect measurement results. These include low light signal perfusion, ambient light, patient movements and high carboxyhemoglobin levels as the most common ones (26; 27).

In spite of these limitations, pulse oximetry is considered as a standard of care for monitoring patients during anaesthesia, which states the clinical importance and validity of the oximeter read SpO_2 and heart rate vital signs (29).

The developed system fully integrates a wireless oximeter (see chapter 3.5.3). The oximetry signal was chosen to be the first one to integrate due to its high clinical importance, wide applicability and due to the fact it was requested by the *Centro Cirúrgico de Coimbra*'s personalized solution (see chapter 2.4).

4.1.2. **Electrocardiograph**

The electrical activity of the heart over time can be recorded by an electrocardiograph thus producing an electrocardiogram (ECG) which is very useful to determine several heart conditions.

At each pulse beat, cardiac cells depolarize and repolarize which results in an electrical current that extends throughout the body. In an ECG, several electrodes are placed at specific locations on the body surface in order to record electrical activity produced at each heat beat. The measured electrical activity does not represent absolute voltages, only voltage changes from a reference baseline. An electrical activity graph over time is then created (ECG) thus allowing clinicians to infer over a patient's heart condition (30).

ECG is considered as a crucial diagnostic tool among clinicians. It is especially useful in providing important information about a patients' heart rhythm, changes in conduction of electrical current from opposite portions of the heart, myocardial ischemia, infarction or increased thickness of heart muscle (30; 31).

Sometimes ECG results can be nearly regular in patients with coronary artery disease or, on the other hand some abnormalities can be present in ECG exams of completely healthy patients. These respectively false negative and false positive conditions constitute known limitations of the ECG technique (32).

Electrocardiograph integration with the developed system was not accomplished however some advances have been made. The sensor device available for ECG integration was Meditech's CardioBlue looping ECG event recorder with Bluetooth. The main features of the device are presented in the following table.

Table 4.1: Meditech CardioBlue main features (33).

Feature	Description
Power Supply	1 AAA alkaline or rechargeable battery
ECG Channels	up to 3 independent bipolar or 5 independent unipolar channels
Sampling	1200 Hz or 600 Hz
A/D resolution	12 bit
Frequency Range	0.05-100 Hz (-3dB)
Dynamic Range	16 mV
Storage Frequency	600 Hz, 300 Hz or 150 Hz
Storage Capacity	total 8 to 60 minutes, depending on quality and lead layout
Connection	Bluetooth
Weight and Dimensions	49 g (without battery) and 66 x 59 x 17 mm

In order to integrate somehow the device into the Vital Signs Monitoring system, its communication protocols need to be known. Meditech does not make communication protocols available to general public therefore some conversations were made with Meditech's quality manager in order to retrieve the required documents. A Confidentiality Agreement was then signed between ISA and Meditech and the communication protocols and technical specifications of the CardioBlue device were supplied and are in the possession of the student.

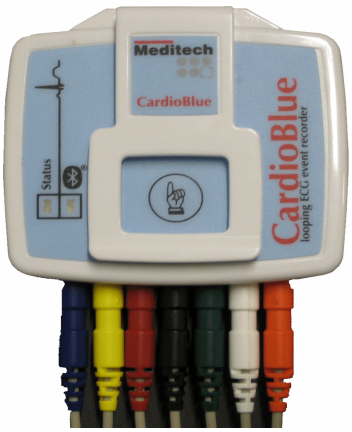


Figure 4.2: Picture of CardioBlue ECG event recorder.

The integration of the ECG recording device into the Vital Signs monitoring system was discarded by three main reasons. The first was the fact that the communication protocols only became available in a late stage of the project's development, thus leaving little time to an additional developing task. The second reason is that *Centro Cirúrgico de Coimbra*'s

personalized solution did not require ECG monitoring which decreased the priority of the ECG device's integration. Finally, the limitation found in the Bluetooth module used in the Transmission Module. As said in the chapter 3.5.2, Free2move's Bluetooth module with Wireless UART allows pairing with only one Bluetooth device at a time. By the time that CardioBlue's communication protocols became available, the system was already integrated with the oximeter which left no space for further sensor integrations.

4.1.3. **Thermometer**

The normal body temperature of a healthy and resting human adult is stated at 37.0 degrees Celsius. However, there is a range in which human body temperature varies according to gender, metabolism, time of day and age (34).

The thermometer is the device used to measure body temperature, which can be obtained in four ways: orally, rectally, axillary and by ear. Each of the four ways has its own typical temperature values.

Body temperature monitoring is important in order to detect as quickly as possible hyperthermia or hypothermia situations. Abnormal body temperature values can indicate disease, injury or pharmacologic activity. Body temperature provides clinicians cues to the detection of infection, inflammation and antigenic responses. It also provides valuable indication to the efficacy of treatment. Continuous temperature monitoring improves body temperature based diagnosis since it provides the analysis of patterns of fever or temperature gradients (35).

In the Remote Vital Signs Monitoring system's context, thermometer did not constitute a big priority. According to the system's architecture, a wireless thermometer suited for body temperature measurement had to be used. Since there was not an available thermometer with the required specifications, a new one had to be developed which led to the abandonment of the temperature vital sign until the remaining phases of the project were concluded.

4.1.4. **Accelerometer**

An accelerometer is a device whose function is to measure acceleration and gravity induced reaction forces. Market provides many types of this kind of devices however 3-axis miniaturized accelerometers are shown to be the most suitable for clinical applications (36).

One of the most used clinical applications for accelerometers is fall detection. Falls are one of the leading causes of injuries among elderly and physically disabled people. A fall

detection system would then allow faster assistance in case of unconsciousness thus minimizing the risk of severe injuries (37).

More recently, some advances have been made in order not to detect, but to prevent falls. In order to achieve fall prevention, subject's accelerometrical activity has to be constantly monitored. Several algorithms based on wavelets, frequency and computational intelligence are applied to accelerometer data in order to identify gait and balance evaluation thus detecting people's ability to walk (38; 39).

Accelerometer's integration into vital signs monitoring systems is then a new and important clinical aid in the prevention of severe injuries.

There were made some advances regarding the accelerometer integration on the Vital Signs Monitoring system. A small system for test purposes was then developed comprising a Texas Instruments ADXL330 accelerometer and an acquisition system. The acquisition system consisted on the PICDEM Z demo board (see chapter 3.2) with some adjustments. It was supported by a PIC microcontroller which took care of A/D conversion, acceleration value calculation and system's extern communication. A/D conversion was made through the PIC's internal ADC with a 10bit resolution and communication was made through RS-232, using the demo board's installed transceiver and nine pin plug. The purpose of this parallel work was to evaluate the behaviour of ADXL330 and PIC's ADC integration in a continuous monitoring scenario.

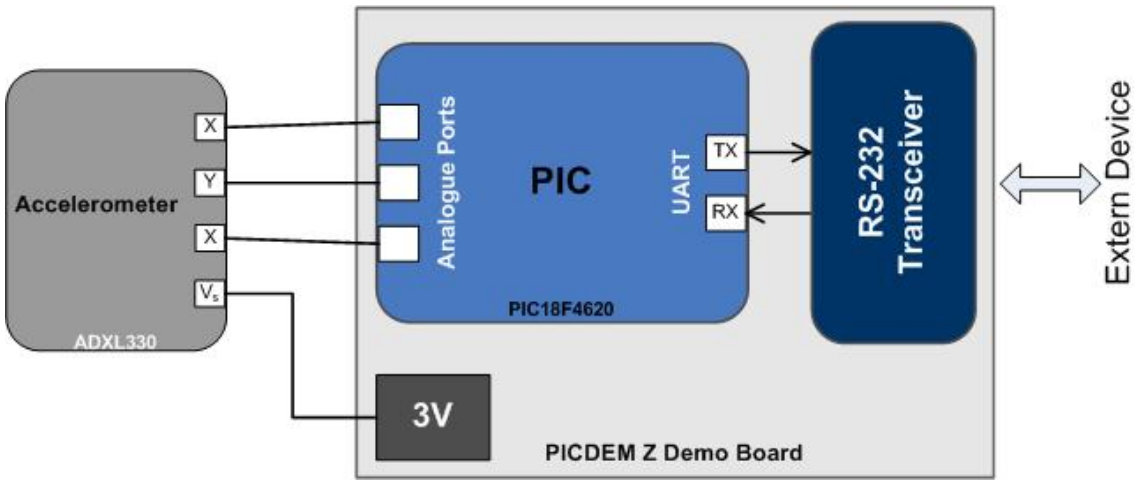


Figure 4.3: Illustrative scheme of the acquisition system's architecture and accelerometer connections. Note for the three analog channels (one for each axis) connected to separate PIC analog ports. A 3 Volt power, from the demo board is supplied to the accelerometer's Integrated Circuit.

A firmware program was then developed in order to handle both the RS-232 communication with the extern device and the A/D conversion of the analog voltage values from the accelerometer. The three channel conversion is made sequentially and converts not only analog voltage levels into digits but also voltage digital values into acceleration in g units. Converted values are then transmitted through UART module and RS-232 transceiver into an external device with minimum delay.

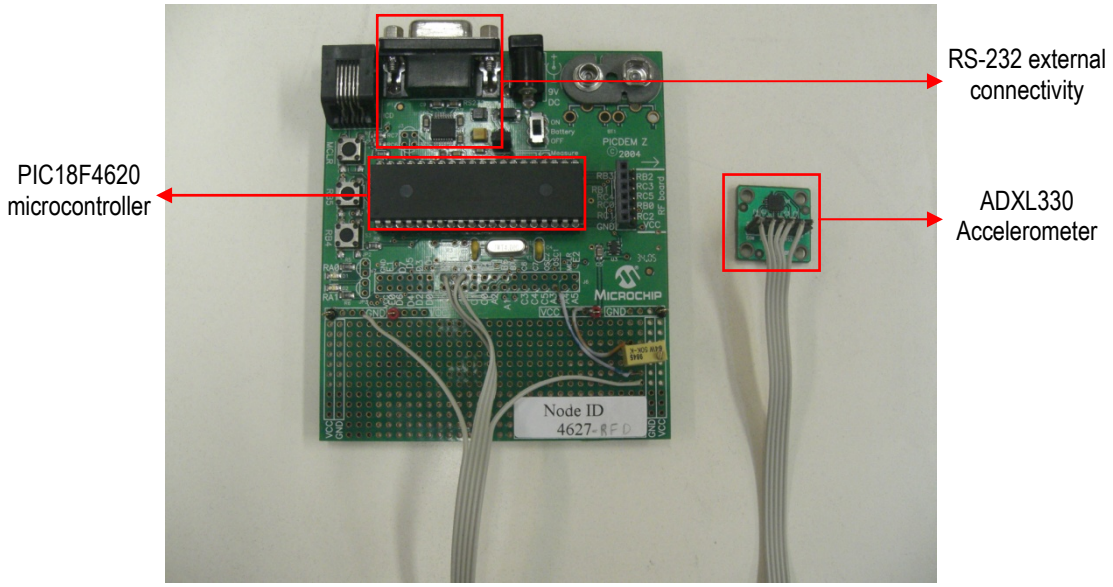


Figure 4.4: PICDEM Z demo board acting as acquisition system of the ADXL330 accelerometer.

In order to allow proper accelerometer data visualization, a PC software application was used whose function was to collect data from RS-232 serial port and show it in a real-time graph with the 3-axis simultaneous accelerometer information. This software piece was made by Rafael Simões (see chapter 2.2).

The conclusions regarding this small parallel work were based only in visual and superficial tests. Entire's system sensitivity was very good as little accelerometer variations were very well noticed in the visualization software graph. Also, system's repeatability was acceptable since measurements made with the accelerometer at rest resulted in imperceptible graphical changes. Additionally, system's accuracy was confirmed to be also very good, as with a horizontal displacement of the accelerometer, the value of one g was obtained along z axis. Despite all the optimistic conclusions, in order to obtain well-founded conclusions, deeper tests need to be carried on to support statistical data treatment.

4.2. Short-range wireless communication

The purpose of short-range wireless communication technologies in the system's context is to provide vital data transmission from the patient's location into the Transmission Module, located nearby. The Wireless Personal Area Network (WPAN) concept seems to be the most suited to this particular application since it interconnects devices around a person's workspace wirelessly. In order to achieve optimum flexibility, security and compatibility, standards based solutions must be used. The most typical technologies in a WPAN are Bluetooth and Zigbee, also referred by IEEE 802.15.1 and 802.15.4, respectively (40; 41).

The following topics try to give a brief comparison on Bluetooth and Zigbee technologies in order to provide enough information to conclude on the most suited technology to apply on the Remote Vital Signs Monitoring system.

4.2.1. **Technologies comparison**

The following table resumes the most important features of both Bluetooth and Zigbee WPAN technologies.

Table 4.2: Features' overview on Bluetooth and Zigbee technologies (42; 43; 44).

Characteristic	Bluetooth	Zigbee
Standard	IEEE 802.15.1	IEEE 802.15.4
Typical Range	1 - 100 meters	10 - 40 meters
Data throughput	1 Mbps	20, 40 and 250 kbps, depending on the operating frequency
Latency	>3s (if changing from sleep to active state)	15ms
Power profile	Low power	Very low power
Operating Frequency	2.4GHz ISM	868 MHz, 902-928MHz, 2.4GHz ISM
Number of devices per network	8	2 to 65000
Network Topology	Piconet, scatternet	Star, peer-to-peer
Cost	Low	Very Low

Besides technical specifications overview, some other considerations must be taken into account while studying the more suitable solution. A research carried out during the early phase of the project indicated that most wireless vital signs sensors' manufacturers enabled their devices with Bluetooth technology regardless of Zigbee. In addition, among hardware modules that provide short-range wireless connectivity, Bluetooth was the most used technology. Documentation support is then more extensive for Bluetooth than Zigbee. These reasons state Bluetooth as the most widely used technology for WPAN.

4.2.2. **Final Considerations**

As it was seen in the previous topic, Zigbee communication offers lower power consumption and better communication flexibility with an increased maximum number of devices per network and lower latencies, at lower costs. The inconvenience is the less wide availability, in particular among the vital signs sensors manufacturers, which turns Zigbee into a powerful but not yet disseminated technology.

In the Vital Signs Monitoring system's particular case, Zigbee could be an interesting solution assuming that the vital signs sensors to integrate did not have built-in wireless capabilities. In this case, Zigbee would possibly be the most suitable solution to connect wirelessly both the sensors and the Transmission Module, due to its very low power profile and increased flexibility. As stated in chapter 2.4, the planning was to integrate market available sensors with built-in wireless capabilities. Due to the unavailability of Zigbee enabled vital signs sensors face to Bluetooth enabled ones, Bluetooth was the obvious choice to integrate system's WPAN.

4.3. Wide-Range Communication

After being transferred from sensors into the Transmission Module by means of a short-range wireless technology, vital signs' data must be sent into a remote Datacenter in order to be stored in the database and be available for remote access. Datacenter's location is uncertain and strongly depends on the system's implementation environment. Once in hospital environment, Datacenter should be located in the same building in which the vital signs monitoring occurs while in Telehomecare environment Datacenter's location could be somewhere in the entire world. The

wide-range communication technology used strongly depends on the Datacenter's location, which in turn depends on the system's integration environment.

Due to the inherent segmentation of wide-range communication technologies into both hospital and Telehomecare environments, the requirements analysis will be done separately into each of the system's integration environments.

4.3.1. **Hospital Environment**

When in hospital environment, it makes sense that the Datacenter and the several Transmission Modules are on the same network. This network is often called a Local Area Network (LAN) and can comprise just one building or several buildings in a health compound.

Both wired and wireless technologies can be used in a LAN. Regarding wired communication technologies Ethernet is most frequently used while Wi-Fi is the correspondent technology regarding Wireless LANs. Next, it will be given a brief review on each of the referred LAN communication technologies in order to discuss the suitability of each technology on the Vital Signs Monitoring system's context.

Ethernet

The first Ethernet standard was published in 1980 and since then it suffered several modifications until it became the official Ethernet standard, also called IEEE 802.3 (45).

Ethernet standard, as every communication standard, provides a set of rules on which manufacturers must base themselves in order to develop devices that are fully and automatically integrated into Ethernet based LANs.

The rules stated by IEEE 802.3 standard are related to both physical and data link layers of the OSI Model which respectively define the physical medium and the way data are transmitted from its source to its destination (45).

Despite the first Ethernet standard applied for a 10 Mbps transfer rate, most nowadays' LANs are equipped with hardware that allows a more recent 100 Mbps standard. Modern versions of Ethernet hardware devices already provide data rates up to 1 or 10 Gbps; however they are not yet disseminated (45).

The advantages of using Ethernet communication technology are its wide infrastructures availability, low cost, backward compatibility and high reliability. In addition, it still is the LAN technology that provides higher transfer rates and lower latencies (45; 46).

On the other hand, Ethernet based devices inside a LAN will have reduced mobility face to Wi-Fi enabled ones.

Again, contextualizing into the Vital Signs Remote Monitoring system, Ethernet technology was chosen to be the first one to implement for several reasons. Firstly it still is the most commonly used LAN technology among Portuguese healthcare units. Additionally, it already existed at ISA a hardware piece that confers Ethernet capabilities into the Transmission Module, thus making the developing process easier and finally, *Centro Cirúrgico de Coimbra*'s personalized solution was to be inserted into an Ethernet based LAN which increased the priority in developing an Ethernet enabled system (47).

Wi-Fi

Wi-Fi stands for Wireless Fidelity and consists on a wireless LAN technology that follows the IEEE 802.11 international standard rules (48).

The purpose of the standard is to provide wireless connectivity inside a LAN with high reliability, high throughput, continuous connection and wired network integration. IEEE 802.11 consists of three layers: logical link control, media access control and physical layers. Logical link control layer provides an interface to higher OSI Model layers and error and flow control functions while media access control layer handles the addressability. Physical layer, in turn, comprises several standard operating schemes resulting in different bandwidths and degrees of stability (see Table 4.3) (48; 49).

Table 4.3: IEEE 802.11 physical layer standards (49).

Standard	Release Date	Frequency operation (MHz)	Maximum allowed data rate (Mbps)
802.11	1997	2.4 to 2.4835	2
802.11a	1999	5.15 to 5.35 or 5.725 to 5.825	54
802.11b	1999	2.4 to 2.4835	11
802.11g	2003	2.4 to 2.4835	54

The advantages of using Wi-Fi over Ethernet resume to its ability to confer mobility into the devices inside the LAN that support this kind of technology. Moreover, it represents a good solution when building a LAN over difficult to wire structures (50).

On the other hand, Wi-Fi is still a technology in a development process. Some issues like range limitation, susceptibility to interferences and security turn it into a less wanted solution in

low failure tolerance scenarios. In addition, data transfer rates are lower than the ones achieved with Ethernet and most of the times the practical value are far from the theoretical ones (43).

Wi-Fi technology was not implemented in the present's year project. The main reason is that Ethernet implementation had a bigger priority, for the reasons presented in the previous topic. However, some steps were made in order to allow a future implementation of this kind of technology in the Transmission Module. Besides the design of the hardware piece that provides Wi-Fi connectivity to Transmission Module by Tomé Matos (see chapter 2.2), some aspects related to Firmware Design were taken into account in order to make integration of the new Wi-Fi hardware piece easier.

4.3.2. **Telehomecare Environment**

As stated above, in a Telehomecare environment, the communication between Transmission Module and Datacenter is uncertain, as the location of the Datacenter is unknown. It is not achievable to integrate every home in the same LAN as the Datacenter however it is very easy to integrate an electronic device into the most Wide Area Network (WAN) called Internet. This way it is possible to transmit data from a device to another one with unknown geographical location. There are several technologies that provide a gateway to the Internet however GSM/GPRS is the only one that does not require infrastructures' installation. GSM/GPRS infrastructures are already installed all over the place, resulting in an almost complete geographical coverage thus allowing Internet access anywhere and anytime. Next, it will be given a brief overview on the GSM/GPRS technology in order to allow a better understanding of its role in the Telehomecare environment Vital Signs Monitoring system as well as its limitations and main advantages.

GSM/GPRS

GSM, which stands for Global System for Mobile Connections, is a mobile telephony network based on the cellular concept. This means that several radio terrestrial cells provide coverage to compatible terminal devices (51).

Besides the popular voice services, GSM also supports data services. In Circuit-Switched data, a dedicated data call similar to a landline modem connection is created in order to provide a data throughput rate of 9.6 or 14.4 Kbps. Additionally, there is the Short Message Service (SMS) that allows sending and receiving short text messages of approximately 160 characters between GSM enabled electronic devices (51).

In order to improve GSM data services, an overlay service was created, called General Packet Radio Service (GPRS). This service operated onto existent GSM networks, providing faster data transfer rates, an “always on” connection, improved connectivity and, most important, broadband application support. This last feature allows IP applications to run over GPRS networks (51).

Due to its almost global coverage, mainly in developed countries, GSM/GPRS appears to be the most suited solution to provide wide-range connectivity on environments with any other existent communication infrastructures. On the other hand, the appeal on GSM/GPRS technologies requires subscribing into extern service suppliers who will obviously charge according to the exchanged data amounts (52).

GSM/GPRS technology was not implemented in the Vital Signs Remote Monitoring system throughout the present year. The priority was to integrate, in first place, communication technologies that do not require external service suppliers which temporarily put aside GSM/GPRS. However, it exists, at ISA, a hardware piece called GSM/GPRS socket that can be integrated into the Transmission Module thus enabling the possibility to communicate via this technology. Of course this integration requires additional firmware developing.

4.4. Transmission Module Capabilities

The purpose of this section is to discuss the requirements of system’s Transmission Module besides the, already presented, wide and short range communication capabilities. The topics approached during this section include data storage, clock and calendar, direct connectivity, portability and modularity requirements.

It is very important to know in advance all the requirements that are needed to implement the system with the desired functions. A limitation found in the development stage of the project can lead to improper adaptations and loss of already done work.

As said in chapter 2.4, the hardware that was used in system’s Transmission Module belonged to an ISA project and was already designed, however it is expected that at the end of this section it can be concluded that the used hardware is well suited to play Transmission Module’s role.

4.4.1. **Data Storage**

When in a normal operation monitoring session, vital signs' data are sent from Transmission Module to Datacenter as soon as it is acquired. However, connection between these two devices can be temporarily lost and vital signs acquisition should not be compromised. Acquired data must be internally stored until connectivity to Datacenter is re-established in order to prevent data loss.

Furthermore, Transmission Module may have the capability of storing an entire monitoring session of several hours. This is of special importance when information in real-time does not have increased importance and the measuring environment does not provide Datacenter connectivity. This way historical vital signs' data can be accessed afterwards.

According to the two distinct presented situations, the amount of data storage required will differ. In case of only loss of data prevention due to communication issues is considered, small storage capabilities will be required. On the other hand, for offline monitoring capabilities, Transmission Module should be able to store several hours of vital signs' acquisition data. Additionally, memories must be able to retain data in power loss situations, in other words, non-volatile memories must be used.

The hardware that was used in the system's Transmission Module contains three non-volatile memories. One is a 128Kbyte EEPROM and the other two are Flash memories with 4Mbit and 16Mbit respectively.

Considering the developed Vital Signs Monitoring system, the amount of data storage was more than enough to implement both data loss prevention and offline monitoring; but only oximeter data was acquired. Data storage requirements will increase after other vital signs' data are considered; nevertheless Transmission Module's hardware is prepared to include a microSD card holder that will strongly increase its data storage capabilities.

4.4.2. **Clock and Calendar**

After being acquired, each vital sign measure must be attached to time and date information. This is done in order to provide clinicians the exact time information that a certain health condition occurred. The attachment of time and date information to each vital sign measure must be done with the minimum delay possible so that time information is as close as possible with the time instant that vital sign data was acquired.

The time and date attachment process must, then, be done within Transmission Module so that as soon as data arrives from sensor, time and date information can be attached.

A Real Time Clock Integrated Circuit (RTC) is contained within the Transmission Module's hardware with the only function of providing exact time and date values. In addition, this device continues to track time and date when no power is provided recurring to a backup battery, thus allowing the conservation of time and date information when Transmission Module is turned off. The minimum amount of time that the used RTC device is able to resolve is one second.

4.4.3. **Direct Connectivity**

Although the final destination of the acquired vital signs' data is the Datacenter, there may be situations where a direct access to the Transmission Module is required. The Transmission Module should be able to provide stored data downloading into a PC or PDA or even to make a real-time monitoring session through a direct connection into the PC or PDA. This way, direct connectivity helps surpassing situations of wide-range communication failure or inexistence.

The used Transmission Module allows USB direct connectivity which is probably the most suited technology to directly connect two electronic devices. It is present on most modern PCs and PDAs; it is a reliable technology and allows high data rates and Plug and Play configuration.

4.4.4. **Portability**

Despite being a device which does not require increased mobility, there may be situations where Transmission Module must provide a certain degree of portability. These include, for instance, patient transfers, patient mobility beyond sensor's range and temporary power losses. Once the module is placed in a particular spot, the patient can recover its mobility, provided by the short-range communication technology. Three particular features were taken into account concerning Transmission Module's portability which are dimension, weight and power autonomy; they will be discussed next.

Dimension

Transmission Module's dimensions are 168 mm (L) x 107 mm (W) x 35 mm (H) which are not suited to a device constantly worn by the patient. However Transmission Module is to be

moved from different places and not to be constantly transported by the patient which turns its dimensions well suited to its function.

Weight

The weight of the Transmission Module, considering both complete electronic part and plastic box, is approximately 300g. Again, like it was discussed in the previous item, the weight value is not suited to a permanent portable device but it is a reasonable value considering Transmission Module's function.

Power Autonomy

Power autonomy is a required condition in portable devices of any kind. Transmission Module must be able to maintain its operation when not connected to the main power supply. The Module contains an internal rechargeable battery that will take place every time the module is disconnected from its power supply. The battery will automatically recharge when the module is reconnected to the power supply.

Tests made in battery powered measure operation stated that the maximum autonomy of the Transmission Module is approximately two hours. Again, concerning the Module's function, this autonomy is considered enough; however it can be easily increased, if needed, applying a bigger capacity compatible battery.

4.4.5. **Modularity**

Modularity stands for the device's ability to integrate other hardware modules in order to improve its capabilities. Two main requirements were identified that increase a device's modularity. The first is the presence of predetermined fixation points, called headers or connectors that allow the integration of socket devices. These headers must provide access to the required pins. The other requirement lies in the microcontroller's capabilities. The main processor must have sufficient processing power and enough peripherals in order to manage all the hardware modules properly.

The used Transmission Module has increased modularity. It contains three socket headers and a header to a specific GSM modem. The three headers comprise a Bluetooth/Radio socket, a GSM/GPRS or Wi-Fi socket and an Ethernet socket. All four headers are connected to the main microcontroller thus confirming its capabilities. Additionally, it runs at 40 MIPS which is a considerable processing power among microcontrollers.

The Transmission Module that was used throughout the system's developing stage had two sockets mounted which were Ethernet socket and Bluetooth socket thus enabling the Module with these two communication technologies.

Chapter 5

5. System Specifications

Throughout this chapter system architecture as well as specific information on each of the system's elements will be described. This chapter is of special importance since it provides the reader detailed information on each component, helping him to fully understand the Firmware Design chapter. Additionally, it gives a good perspective on the work that had to be carried out before the firmware development stage of the project.

The chapter is divided into four main sections. The first one provides an overview on the entire system's architecture and the three following ones provide details on each of the system's units which are the sensor, Transmission Module and Datacenter.

This chapter, as well as the following one, Firmware Design, concerns only the developed system, which comprises *Centro Cirúrgico de Coimbra*'s personalized solution. As it was previously mentioned, this implies the use of Bluetooth and Ethernet as short-range and wide-range communication technologies, respectively and oximetry as the only measured vital sign. This way, from now on, when referring to Vital Signs Remote Monitoring system, only the personalized developed solution will be taken into account.

5.1. System Overview

Vital Signs Remote Monitoring system can be divided into three main levels, according to the information pathway which are the Sensor, the Transmission Module and the Datacenter. The scheme presented in Figure 5.1 tries to exemplify the relations between the three levels as well as the communication technologies that connect each one of them. Bluetooth module limitations (see chapter 3.5.2) imply one Transmission Module per sensor which means that each patient has its own Transmission Module. All Transmission Modules communicate via Ethernet into the same Datacenter that is connected in the same LAN.

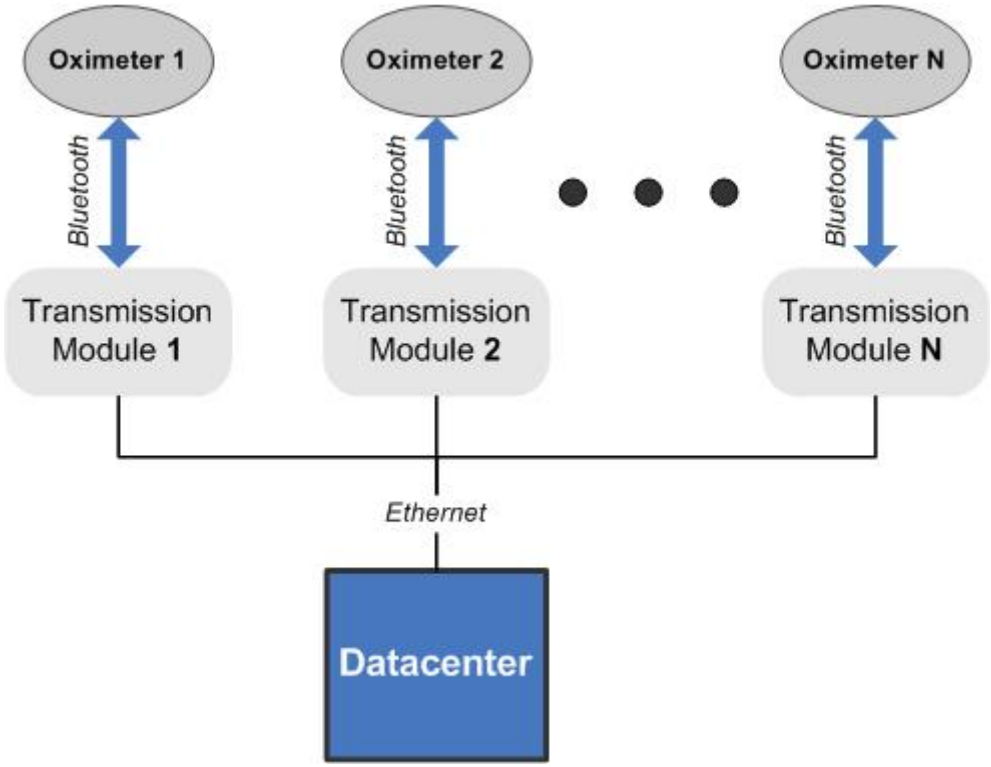


Figure 5.1: Illustrative scheme of the Vital Signs Remote Monitoring system's overall architecture. Note for the three horizontal levels regarding the information pathway.

The information pathway starts at the sensor which acquires and processes the oximetry signal. Oximetry values are then transmitted to its correspondent Transmission Module. Each Transmission Module is responsible for storing the received oximetry values and transmitting them via Ethernet with the minimum delay into the Datacenter, which is the final destination of the information pathway.

5.2. Sensor

The sensor used in the developed system was Nonin 4100 Pulse Oximeter with Bluetooth which provides Heart rate and SpO_2 vital signs measurement. Bluetooth related details of the oximeter were previously presented in chapter 3.5.3.

As it was previously said, the device acts as a Slave, therefore the pairing process (see chapter 3.5.1) is handled by the Bluetooth module that acts as a Master. Once paired, a SSP connection is established and information can be transparently exchanged. In order to properly retrieve oximeter's acquired data, its Communication Protocol must be known in advance. The following considerations concern the oximeter's communication protocol.

According to the manufacturer's document, referenced as (25) and which comprises the device's communication protocol, in order to start measuring SpO_2 and Heart rate vital signs, a command must be sent to the oximeter. This command consists on a predefined string, "D8". As said, in the communication protocol document, this command will put the oximeter in the measuring mode with *Serial Data Format 8*. In this mode, at each second, one measure will be sent by the oximeter to its Master device which comprises 4 data bytes. Each measure consists on the average value of four oximeter acquisitions. Table 5.1 contains the description of all the four bytes that comprise the information of one measure. A deeper description can be found in Annex A - Table: A.1.

Table 5.1: Description of the 4 bytes that comprise one measure's data packet (25).

	Name	Description
Byte 1	STATUS1	Information on the signal quality and 2 MSB of Heart Rate value
Byte 2	HEART RATE	7 LSB of Heart Rate value
Byte 3	SpO_2	SpO_2 value (7 bits)
Byte 4	STATUS2	Information on the measuring mode and battery status

The parameters of the SSP connection created between Bluetooth oximeter and Bluetooth module are defined by the oximeter and are presented in the following table.

Table 5.2: Serial communication parameters (25).

Bits per second	Data bits	Parity	Stop Bits	Flow Control
9600	8	None	1	None

5.3. Transmission Module

The Transmission Module is the main component of the Vital Signs Monitoring system. Its main role is to act as a bridge between the patient's oximeter and the Datacenter. Sensor data retrieving, Bluetooth connection handling, data storage, time information attachment and Datacenter communication handling are one of the most important tasks carried out by the Transmission Module, thus making oximeter data transference a transparent operation to the Datacenter.

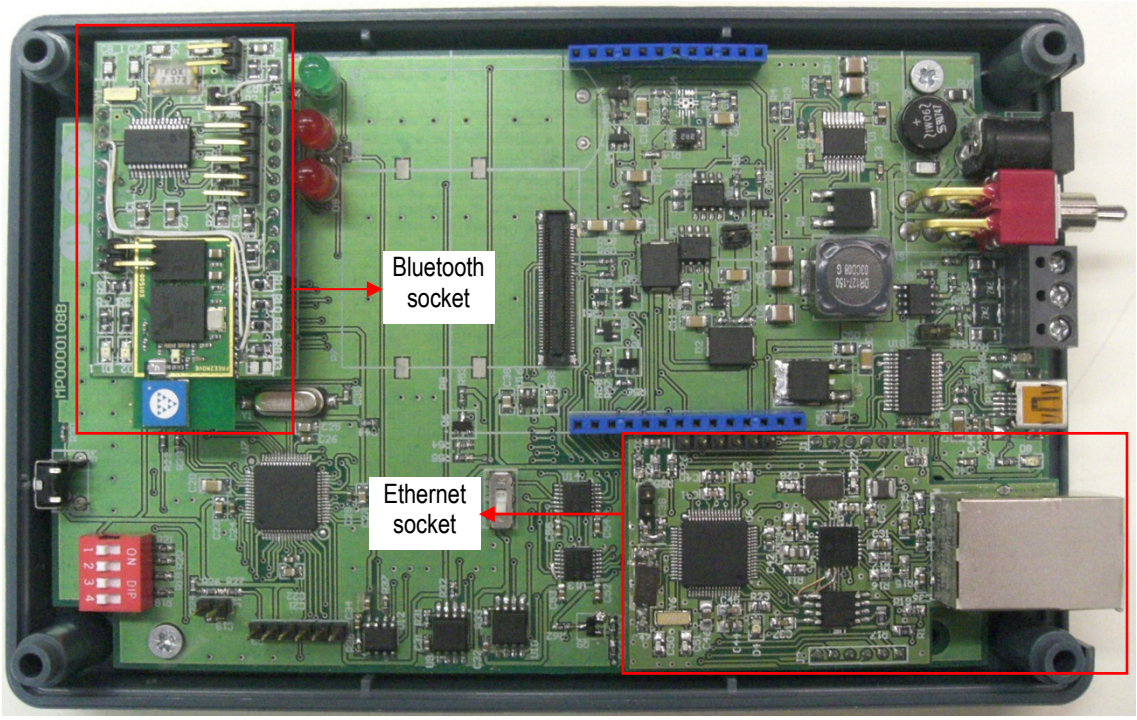


Figure 5.2: Picture of the Transmission Module's prototype. Note for both the Bluetooth and Ethernet sockets inserted in the Main Board.

As the reader may have realized in the previous chapters, the Transmission Module that was used to accomplish the Vital Signs Monitoring system is a hardware piece composed by a main board and two socket devices. One handles Bluetooth communication and the other handles Ethernet communication and TCP/IP connections over the LAN. The following scheme presents an overall illustrative view of the Transmission Module with both socket devices as well as the communication interfaces that connect each one of the sockets into the Main Board. Figure 5.2 provides a real perspective on the Transmission Module's prototype as well as socket's location throughout the Main Board.

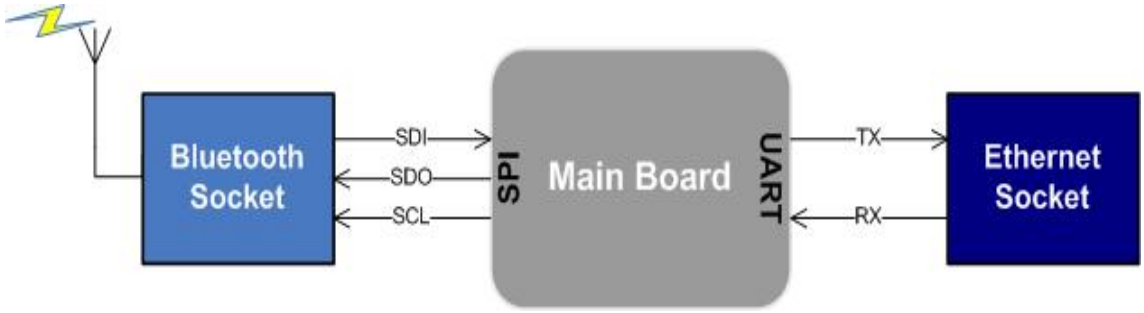


Figure 5.3: Illustrative scheme of the Transmission Module's overall architecture. Note for the serial communication protocols used to connect each socket to the Main Board.

Next, it will be given detailed specifications on each socket as well as on the Main Board, including their internal components. As it was said in the beginning of the chapter, the description is of special importance in order to fully understand the Firmware Design.

5.3.1. **Bluetooth Socket**

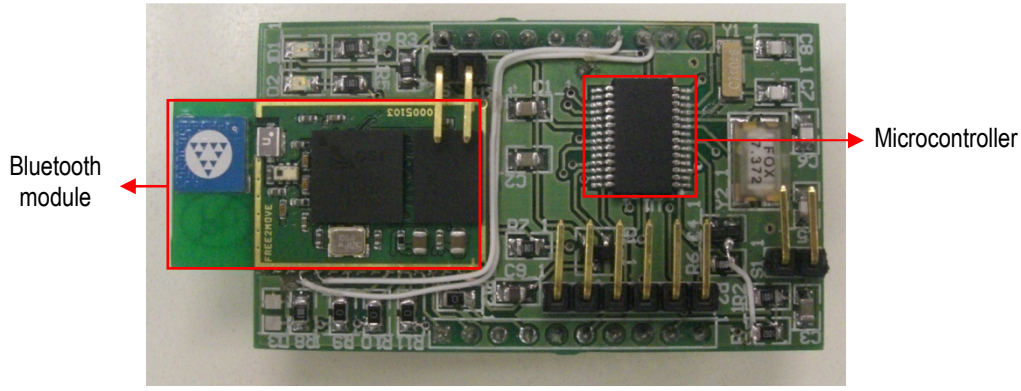


Figure 5.4: Picture of the Bluetooth socket's prototype. Note for both the Bluetooth module and microcontroller which are the main components of the socket.

Bluetooth socket is the hardware piece responsible for handling Bluetooth connections and processing oximeter periodical data. Its two main components are the Bluetooth module and the microcontroller. The microcontroller is the central component since it acts in two different sides. First, it handles Bluetooth communication through the Bluetooth module, and second, it handles data transmission to the Main Board. This way, oximeter's data can be transparently retrieved by the Main Board.

The following figure helps the reader to understand Bluetooth socket's architecture. Figure 5.4 provides a real perspective on the socket as well as its main component locations.

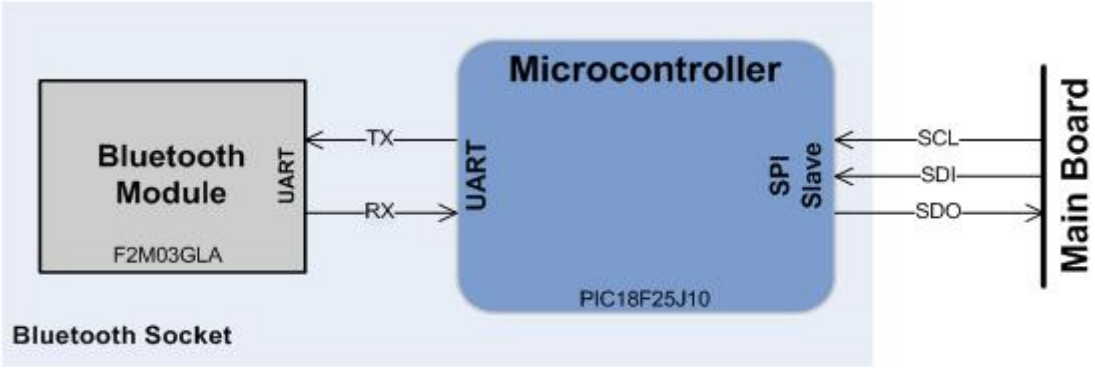


Figure 5.5: Illustrative scheme of the Bluetooth socket's internal architecture. Note for the serial communication protocols used to connect microcontroller into both Bluetooth module and Main Board.

The following topics provide details on each of the Bluetooth socket's presented components. These details comprise microcontroller specific features that were taken into account while designing firmware, and Bluetooth Module's communication protocol, required to control Bluetooth connections.

Microcontroller

The microcontroller model integrated in the Bluetooth socket is the PIC 18F25J10. This microcontroller has 8bit architecture (see chapter 3.2) and provides the developer UART and SPI peripherals. Additional microcontroller's features are presented in the following table (53).

Table 5.3: PIC 18F25J10 microcontroller's main features (53).

Feature		Value
Program Memory	Flash (bytes)	32K
	Number of single word instructions	16384
RAM Memory (bytes)		1024
Number of I/O pins		21
Primary Oscillator Frequency (MHz)		7.3728
Operating Voltage (V)		3.3
Number of Timer units		2

While developing firmware, microcontroller programmer must take into account the configuration of some basic parameters. These include, in the current case, FOSC (Device Output Frequency), F_{CY} (Device operating frequency) and BRG (Baud rate generator). FOSC value is obtained after the PLL multiplier. In the case of the PIC18F microcontrollers PLL multiplies the oscillating source by four, if enabled. In turn, F_{CY} is obtained dividing by four the value of FOSC. Baud rate generator is the PIC microcontroller's register that configures UART baud rate. More specifically it controls the period of a free running timer. According to the PIC's data sheet, the formula that calculates BRG value is $BRG = {FOSC}/{16 \times Baudrate} - 1$ (53).

The PIC microcontroller must handle both SPI and UART peripherals in order to control Main Board and Bluetooth module communication, respectively. In case of SPI communication Bluetooth socket's microcontroller acts as a Slave device while the Main board acts as the Master device.

Despite handling SPI and UART peripheral, Bluetooth socket's microcontroller must handle some additional I/O lines. These include one line to turn Bluetooth module's power ON or OFF and other line to set/unset the same module in a reset state.

Bluetooth Module

As it was previously said, the Bluetooth Module that enabled Transmission Module's Bluetooth connectivity is Free2Move F2M03GLA. Bluetooth related information as well as technical specifications of the module can be found in chapter 3.5.2. In order to the programmer develop firmware routines that handle the referred module, operation as well as communication protocols must be deeply known. This topic's purpose is to give an overview on the referred subjects.

The module's operation is based on two distinct modes. One is called Host Controlled Mode (HCM) and allows the host (microcontroller) to issue configuration commands as stated in the communication protocol. The other mode is the normal operation mode, in which the module will try to accomplish Bluetooth pairing with the configurations issued during HCM. After Bluetooth pairing is accomplished a serial connection is made between the host and the remote Bluetooth device, so any data that is sent to the module is directly sent to the remote Bluetooth device and data received from remote device is directed to host (21).

Control of the module is made by both input/output lines (PIO lines) and UART configuration commands.

In order to control and monitor Bluetooth connectivity two PIO lines are used, as stated in Wireless UART firmware version 4 protocol. One line is an output and provides information on Bluetooth connection status and the other one is an input and allows host device to disconnect or prevent Bluetooth connections (see Table 5.4). These lines allow microcontroller to know if pairing/unpairing occurred and to set/unset Bluetooth module in hold. Detailed description on the Bluetooth module's PIO lines can be found in the following table (22).

Table 5.4: Detailed description of the PIO lines used to control Bluetooth connections (22).

Line	Signal Direction	Active State	Description
PIO2	Input	High	Disconnects or prevents Bluetooth connection
PIO3	Output	High	Bluetooth connection status

Once entered HCM, there are several commands that can be issued. The commands used to accomplish system's development were:

- Configuration commands;
- Software/Hardware reboot commands;
- Inquiry commands (search for Bluetooth devices in the neighbourhood); and
- Pairing and security commands (device authentication);

These commands and their respective responses are sent through UART interface at a factory predefined baud rate of 38400baud, 8 data bits and one stop bit.

5.3.2. **Ethernet socket**

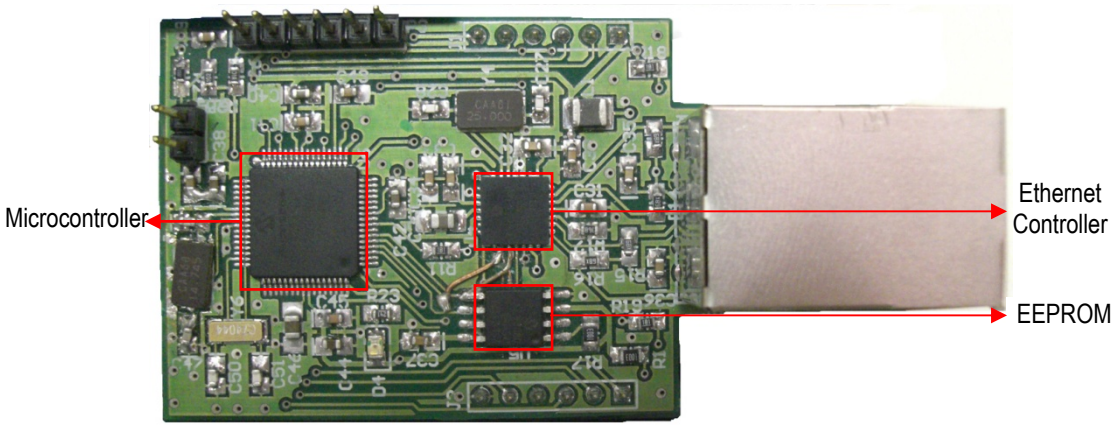


Figure 5.6: Picture of the Ethernet socket's prototype. Note for the Ethernet controller, EEPROM memory and microcontroller which are the main components of the socket.

Ethernet socket is the hardware piece responsible for handling Ethernet TCP/IP connections. Its main function is to act as a bridge between the Main Board and the LAN. This implies IP address acquisition, TCP/IP connections handling and packet translation. Its main components are the microcontroller, the Ethernet controller and the EEPROM memory.

In order to accomplish all the required tasks, microcontroller must be able to control several different sides. These include Ethernet controller, EEPROM memory and Main Board interface. This way data can be transparently transferred between Main Board and the extern LAN.

The following scheme represents the several components of the Ethernet socket as well as the communication protocols used to exchange information. Its purpose is to clarify Ethernet socket's architecture and its role within the Transmission Module. Again, Figure 5.6 provides a real perspective on the socket as well as its main component locations.

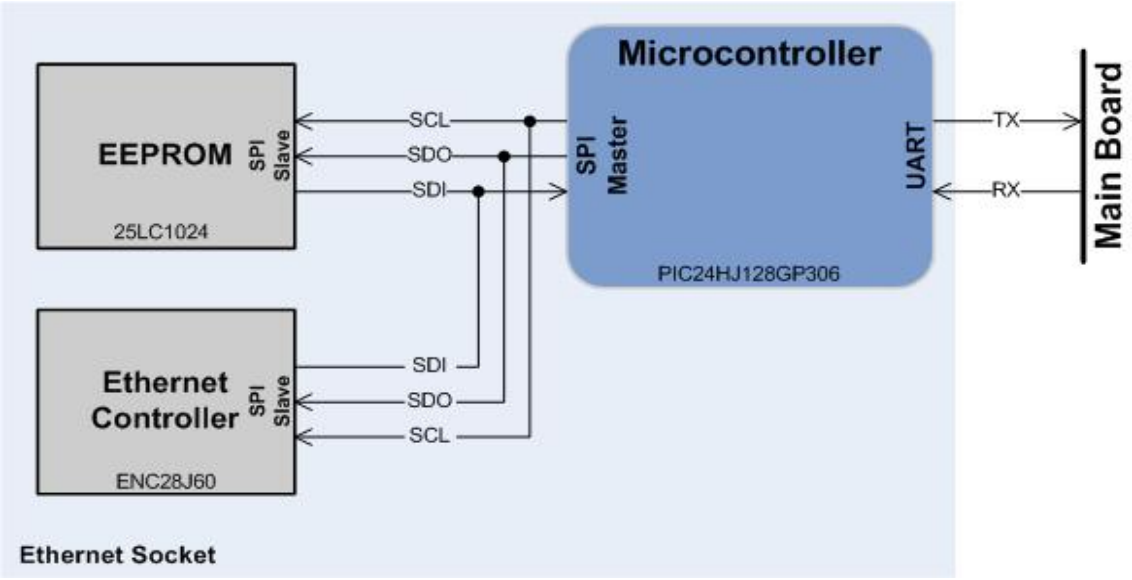


Figure 5.7: Illustrative scheme of the Ethernet socket's internal architecture. Note for the serial communication protocols used to connect microcontroller into EEPROM memory, Ethernet controller and Main Board.

The following topics provide details on each of the Ethernet socket's components. Again, this information is of special importance to fully understand the Firmware Design chapter.

Microcontroller

The microcontroller model used to control Ethernet socket is the PIC 24HJ128GP306 which has a 16bit architecture (see chapter 3.2). This microcontroller provides SPI and UART peripherals to allow control of EEPROM memory, Ethernet controller and Main Board's serial communication. The firmware running in this microcontroller is *Microchip's TCP/IP stack* (see chapter 6.3.1) which requires high processing power. Additional microcontroller's features are presented in Table 5.5 (54).

Table 5.5: PIC 24HJ128GP306 microcontroller main features (54).

Feature	Value
Flash Program Memory (bytes)	128K
RAM Memory (bytes)	16K
Number of I/O pins	53
Primary Oscillator Frequency (MHz)	14.7456
Operating Voltage (V)	3.3
Number of Timer units	9

As it was said before, FOSC, F_{CY} and BRG parameters must be taken into account while developing firmware. In PIC 24HJ's case, PLL multiplication system provides a more flexible frequency specification, but also implies a more complex configuration. The following scheme tries to explain the operation of the PIC 24HJ's PLL system.

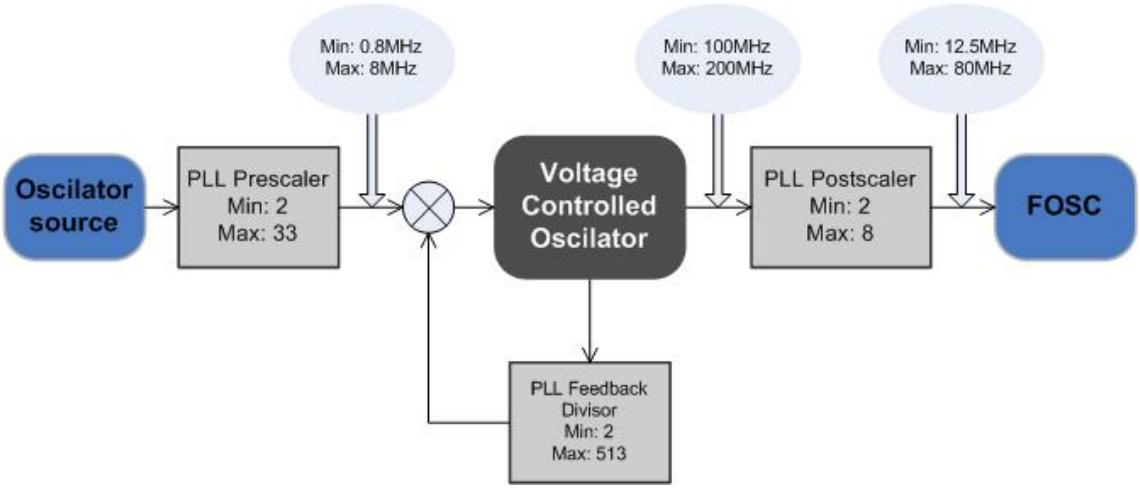


Figure 5.8: PIC 24HJ PLL system block diagram. Note for the minimum and maximum values that must be obtained at the output of prescaler, Voltage Controlled Oscillator and postscaler.

A first, the output of the oscillator source is divided down by a prescaler factor (N_1) ranging from 2 to 33 so that the resultant frequency is in the 0.8-8MHz range. After that, it enters the Voltage Controlled Oscillator which, according with the PLL Feedback Divisor, will multiply the input oscillating frequency by a factor of M that can vary from 2 to 513. Also, this factor must be selected so that the output is in the 100-200MHz range. Finally, the oscillating frequency is divided by a postscale factor (N_2) that can take 2, 4 or 8 values and must provide an output frequency (FOSC) in the 12.5-80MHz (54).

In the PIC24HJ's case, F_{CY} value is obtained dividing by two the value of FOSC. BRG value, in turn, is calculated in the same way as the already presented PIC18F.

EEPROM Memory

The EEPROM memory used in the Ethernet socket is the 25LC1024. As it was previously said, the firmware running in Ethernet socket's PIC is *Microchip's TCP/IP stack*. This firmware code already provides routines to handle this type of EEPROM, so that its operation did not have to be fully understood. However, in order to accomplish proper firmware adaptation into the hardware requirements, some features were studied and are resumed in the following table.

Table 5.6: 25LC1024 EEPROM main features (55).

Feature	Value
Maximum write cycle time (ms)	5
Size (bytes)	1024K
Communication interface	SPI (Slave)

The 25LC1024 acts mainly as a configuration memory within *Microchip's TCP/IP stack*. Its usage resumes to Ethernet related information storage such as default IP address, network name or MAC address. Additionally it can be used to store web pages that will afterwards be provided by the firmware stack.

Ethernet Controller

The ENC28J60 is a standalone Ethernet controller and was used within Ethernet socket in order to enable it with Ethernet connectivity. In this case, similar to the EEPROM memory, the device's operation did not have to be fully understood, as the *Microchip's TCP/IP stack* provides all needed routines to handle this specific device. The following table presents ENC28J60's main features, mostly related to its external connectivity capacities.

Table 5.7: ENC28J60 Ethernet controller main features (56).

Feature	Value
Standard compatibility	IEEE 802.3
Physical Layer integration	MAC and 10Base-T
PIC communication Interface	SPI (Slave)

5.3.3. **Main Board**

Main Board is the Transmission Module's main hardware piece since all the information coming from the surrounding information pathway levels (see chapter 5.1) relies on it. This implies that the Transmission Module's main decision center is located in Main Board's microcontroller.

A real perspective of the Main Board as well as its main component's locations is provided in the following figure.

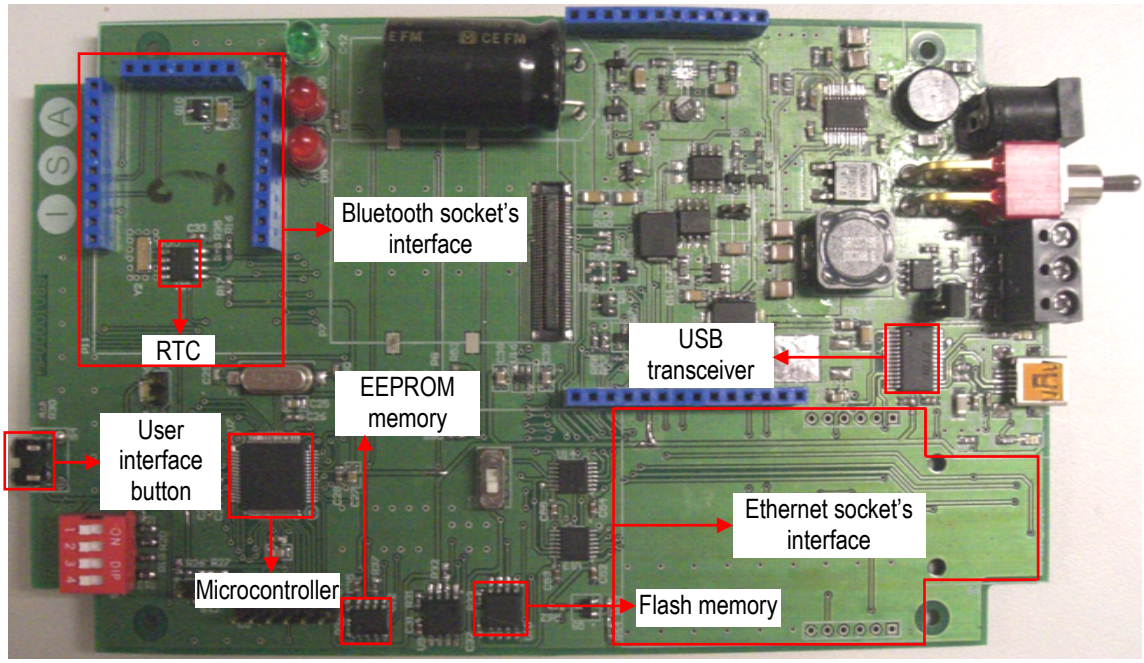


Figure 5.9: Picture of the Main Board's prototype. Note for the both Ethernet and Bluetooth sockets' interfaces, microcontroller, EEPROM and Flash memories, RTC unit, USB transceiver and user interface button.

The main tasks of the Main Board's microcontroller are:

- Handle Bluetooth oximeter data;
- Handle Datacenter communication via Ethernet;
- Accomplish measure's time and date information attachment (see chapter 4.4.2);
- Achieve reliable data storage;
- Handle USB direct connectivity; and
- Handle user interface events.

In order to perform all the presented functions, microcontroller must be able to control several different sides efficiently. These include socket's interface, memories, RTC, USB interface and user interface button.

Figure 5.10 presents a scheme with the several components of the Main Board, its connections and communication protocols used to exchange information.

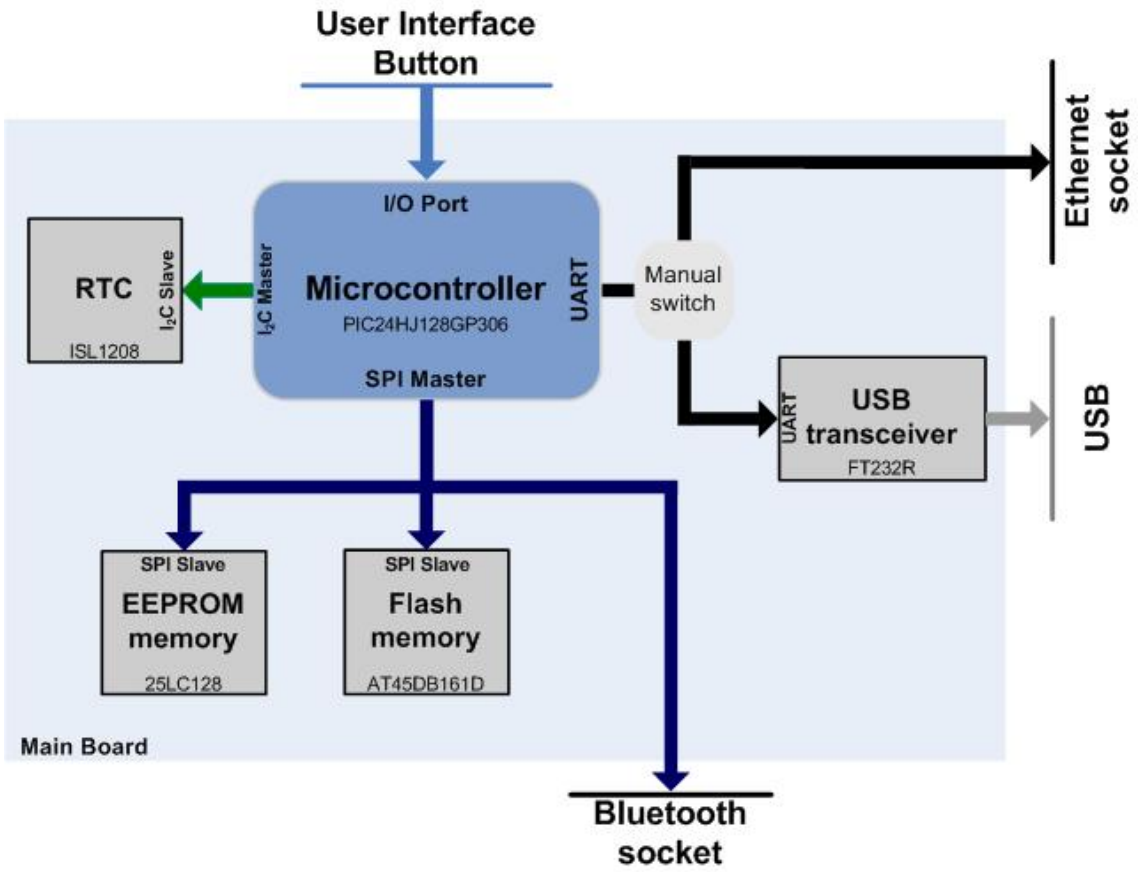


Figure 5.10: Illustrative scheme of the Main Board's internal architecture. Note for the several onboard components (in grey), the communication protocols used to connect microcontroller into both onboard components and extern sockets (dark blue for SPI, green for I_2C and black for UART) and I/O port connection into user interface button (light blue).

The following topics provide details on each of the Main Board's components. Again, the following information is of special importance to fully understand Firmware Design's chapter.

Microcontroller

The microcontroller, that is the core of the Transmission Module, is the same that is used in Ethernet socket (see chapter 5.3.2 - Microcontroller) which is the 16bit PIC 24HJ128GP306. Its high processing power provides enough smoothness while handling all timers, interrupts and main code routines' execution. The peripherals used by the Main Board microcontroller are SPI in support of memories and Bluetooth socket communication, I_2C in order to retrieve information from RTC and UART which communicates with USB transceiver or Ethernet socket's microcontroller. Additionally, it was also used timer units as well as input change notification functions. Input change notification is a function of some I/O ports that allows the microcontroller

to generate interrupts when a pin state changes, which is very useful to detect user interface button events.

Memories

As it was stated before, there were used two non-volatile memories each one with a different purpose which are 25LC128 EEPROM and AT45DB161D Flash memories.

Time attached oximeter measures are stored in Flash memory. Its high capacity makes it more suitable for data storage purposes. On the other hand, EEPROM's simplicity of operation makes it a valid choice for storing configuration variables. The main role of the used EEPROM is to store Flash memory pointer variables, so that when power is lost, the program can know how many data bytes are stored in Flash memory and their location within the memory.

In order to develop firmware routines to read and write information to both EEPROM and Flash memories, their operation and communication protocol had to be fully understood.

AT45DB161D Flash memory is divided into pages 4096 pages of 512 bytes or 528 bytes each. The chosen page size was 528 bytes which increases memory capacity. Entire memory capacity is then 2162688 bytes which is approximately 2Mbytes. Additionally, the memory contains two SRAM *buffers* of 512/528 bytes each, that allows the receiving of data while a page in the main memory is being reprogrammed as well as writing a continuous data stream and save data at the end. Data are read or written through a fast SPI interface which allows frequencies up to 66 MHz (57).

25LC128 EEPROM is a very simple SPI compatible memory. Its 128 Kbits (16384 bytes) are organized into 64 byte pages. Reading and writing routines into different pages was not implemented because less than 64 bytes (page size) were required to be stored (58).

RTC

The RTC device contained in Transmission Module's Main Board is the ISL1208. This device provides real time Clock and Calendar tracking with year, month, day, day of the week hours, minutes and seconds individual registers. Calendar is functional until 2099 with automatic leap year correction. Data retrieving and configuration is made through I_2C interface which allows 400 KHz maximum data transfer rate. One important feature of the ISL1208 RTC device is the ability to continue tracking time in power failure situations, through a small low power battery (59).

Time and date registers depict Binary-coded decimal (BCD) representations. Firmware programmer must take into account this particular feature in order to obtain valid decimal time and date values (59).

USB interface

Transmission Module's USB interface is provided by the FT232R USB transceiver. This device provides the firmware programmer a very simple way to transmit data through a USB port since its operation is fully automatic. In other words, data that enters the device by UART interface is automatically shifted out into the USB port. Entire USB protocol is handled on the chip which means that no specific firmware programming or previous configuration is required. Plug and Play feature is also provided by the transceiver.

However there is one detail that the programmer must take into account. FT232R USB transceiver provides an I/O line that will indicate the microcontroller if an USB cable is plugged or unplugged. Correct handling of this I/O signal is developer's entire responsibility (60).

Data transfer rates range from 300baud to 3Megabaud for microcontroller UART input.

Socket's Interface

As it was previously noticed, Main Board's socket interfaces provides the Ethernet and Bluetooth sockets the communication protocol lines required to connect these sockets into the Main Board's microcontroller.

However, there are some additional I/O lines that are exchanged between Main Board's microcontroller and each one of these sockets.

In the case of the Ethernet socket's interface, besides UART interface lines, there is one general purpose I/O line that can be used to indicate whether there is an active Ethernet link or not.

Bluetooth socket, on the other hand, does not really exchanges additional I/O lines however there is Main Board microcontroller controlled line that allows it to turn Bluetooth socket's power ON or OFF.

User Interface Button

User interface button consists on a pressing button that is connected into a microcontroller's I/O pin and will change the pin's state according to its own state.

In the current system, the purpose of this button is to allow the user to generate an event such as memory reset or sleep state exit.

Details on the operation and handling of this user interface button are presented in chapter 6.4.3.

5.4. Datacenter

Datacenter is the terminal level of the information pathway presented in chapter 5.1. As it was stated before, communication between Transmission Module and Datacenter is made through TCP/IP supported in an Ethernet physical layer.

According to TCP protocol, the roles of each device must be previously defined before any communication attempt. While establishing a TCP connection between two devices each one must have a specific role from TCP server or TCP client. The TCP server must have a known IP address and be listening on a predetermined port. TCP client, in turn, will have to connect to the TCP server's known IP address and port so that the server can be informed of the client's connection establishment intention.

According to the entire system architecture it makes sense that after powered on, the Transmission Module establishes connections as a TCP client and tries to connect into the Datacenter's previously defined IP address and port. This way Datacenter acts as a TCP server can efficiently handle all the connection attempts of every LAN embedded Transmission Module.

Datacenter identifies the several Transmission Modules connected within the LAN through its network name which differs in each Transmission Module.

Chapter 6

6. Firmware Design

Most of the work carried out during the past academic year is experienced along this chapter. Despite being the project's main objective, Firmware Design was the last stage to be performed. Previous chapters can be considered as intermediate steps that aim to a common goal that is the development of the Transmission Module's firmware that performs all the required tasks.

Throughout this chapter, several concepts approached in previous chapters, will be referenced. This way, in order to fully understand present's chapter considerations, the reader must be entirely familiarized with previous chapters.

Besides the actual Firmware Design sections, this chapter includes an additional section that has the purpose to explain the communication protocols defined by the programmer throughout firmware development's stage. This section is presented in the beginning of the chapter and provides important information that allows the reader to fully understand the following sections.

As the reader may have realized, system's firmware design comprises three microcontrollers, each one with a different firmware program. These comprise Bluetooth socket, Ethernet socket and Main Board microcontrollers. Following Communication Protocols Definition section, the chapter is divided into three more sections that present the firmware design of the three microcontrollers.

6.1. Definition of Communication Protocols

Two distinct communication protocols were defined throughout firmware development's stage. One defines the rules of communication between Bluetooth socket's and Main Board microcontrollers. As stated before (see chapter 5.3) SPI is the interface technology that connects

the two microcontrollers. The other defined communication protocol is related with Transmission Module and Datacenter's information exchange over the Ethernet enabled LAN.

Next, it will be described the two defined communication protocols. Additionally, the reasons that stipulated the way the protocols were defined will be explained.

6.1.1. **Bluetooth Socket – Main Board**

The protocol defined to mediate information exchange between Main Board and Bluetooth socket microcontrollers is based on two very simple rules imposed by SPI protocol. One of the rules states that information exchange is reciprocal, in other words, it means that when one byte is clocked out, another byte is simultaneously clocked in. The other rule is imposed by the SPI Master-Slave condition (see chapter 3.2.5 - SPI). According to this condition, communication must be always started by Master device which controls clock line. In the present case, Master device is Main Board's microcontroller while Slave device is the Bluetooth socket's microcontroller (see chapter 5.3).

The philosophy that was thought to be the more suitable for the present case consists on a periodical check controlled by Master device that verifies if there is data available on Bluetooth socket. In case there is, one byte will be retrieved by Master device otherwise nothing is done until the next periodical check.

To implement the referred philosophy, from the Master device's side, two command bytes were defined to initialize conversation. One has the purpose to solicit Bluetooth socket's status and the other is used to request the sending of one data byte retrieved from the oximeter. The following table resumes details on both Master command bytes.

Table 6.1: Main Board SPI command bytes regarding Bluetooth socket's communication.

Name	Value	Description
GETSTATUS	01 (hex)	Requests Bluetooth socket's status
GETDATA	02 (hex)	Requests the Bluetooth socket the sending of one data byte

From the Slave device's side it were also defined protocol rules for the responses that correspond to each of the previously presented Master commands. When receiving any of the two Master commands, Slave device must simultaneously provide one *don't care byte* due to SPI's reciprocal exchange rule. After receiving the entire command byte the Slave device will

recognize it among the two possible options presented on Table 6.1. According to the received byte value the Slave device will take one of three possible actions. In case the received byte is the *GETSTATUS* byte, Slave device will load its SPI register with the status byte. On the other hand, if *GETDATA* byte is received, the oldest data byte available will be loaded into SPI register. However, if the received byte is not comprised in the two defined command bytes, an *ERROR* byte will be loaded into SPI register and recognized by Master device. The following table provides details on the *ERROR* byte, allowing the following discussion on its defined value.

Table 6.2: Bluetooth socket's *ERROR* byte.

Value (hex)	Value (dec)	Value (binary)
AA	170	10101010

The value of the *ERROR* byte was not chosen by chance, but taking into account his binary representation. The value 10101010 (binary) will not be confused by any chance with an oximeter measure's data byte (see Annex A - Table: A.1) and it is very improbable that it is produced by a SPI line instability. Generally SPI line instabilities produce *all ones* or *all zeros* binary representations.

Master device, after sending the command byte will try to retrieve the byte that the Slave device loaded into its SPI register. This is done, sending one *don't care byte* and taking advantage of the SPI reciprocal information exchange. The byte that the Master device will simultaneously receive is the same that the Slave device previously loaded into its SPI register.

The following scheme tries to provide a better understanding of the defined protocol.

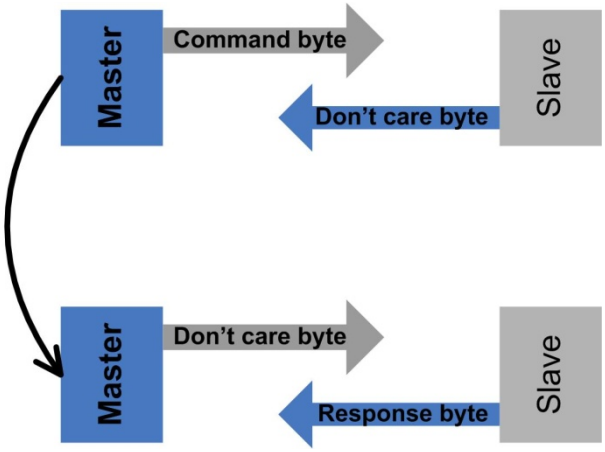


Figure 6.1: Illustrative scheme of the communication protocol defined for data exchange between Bluetooth socket and Main Board. Note for SPI reciprocal and simultaneous byte exchange.

The main advantage of the defined communication protocol is the increased data loss prevention. This capability is mainly due to the fact that when a byte is transmitted from Bluetooth socket, the Master device is already expecting it because it was previously solicited and its reception is properly handled. One disadvantage of this philosophy is the fact there must be sent two bytes in order to retrieve just one data byte. However, for small data amounts this disadvantage does not constitute a limitation.

6.1.2. **Transmission Module – Data Center**

The protocol for the communication between Transmission Module and Datacenter was defined by both Nuno Varelas and Rafael Simões project students. This combination of efforts is due to the fact the each student has a deeper knowledge of his own assignment (see chapter 2.2 - Table 2.2).

Due to system architecture demands it was chosen than the Transmission Module should adopt a passive role towards Datacenter. This means that Datacenter is the element that makes the requests and Transmission Module the element that satisfies these requests. Additionally, it was stated that after any kind of valid request issued by Datacenter, a response should be sent back by Transmission Module whatsoever is the request's resolution.

Three different requests were identified, taking into account system requirements which resulted in three Datacenter request messages. Its description is presented on the following table. Detailed messages' description can be found in Annex A – Table: A.2.

Table 6.3: Datacenter request messages. Note for dual option *Disconnect*_Req message.

Name		Description
Connect Req		Requests connection to the Transmission Module
Start Req		Requests Transmission Module to start sending measure data
Disconnect Req	Option 1	Requests Transmission Module to stop sending measure data
	Option 2	Requests Transmission Module to stop sending and recording measure data and to disconnect

From the Transmission Module's side, as it was stated before, for every valid request message received, a response must be sent back to Datacenter. This way for each of the previously identified request messages, a response was defined. The response message has two main functions. Firstly, it acts as an acknowledge message, in other words, it indicates the

Datacenter that the request message arrived its destination unchanged. The other function regards message content, which indicates if request status is successful or unsuccessful. The following table describes Transmission Module response messages. Detailed messages' description can be found in Annex A – Table: A.4.

Table 6.4: Transmission Module's response messages. Note for dual state *Connect Resp* message.

Name		Description
Connect Resp	State1	Transmission Module successfully connected
	State2	Connection attempt failed because oximeter is unpaired or turned off
Start Resp		*Start Req* was not accomplished. Transmission Module is disconnected
Disconnect Resp		*Disconnect Req* successfully accomplished

It is important to make clear that in order for the Transmission Module to start sending measure's data, a successful *Connect Resp* message must be previously issued to Datacenter. In other words, Transmission Module must be connected to Datacenter. Otherwise, a *Start Resp* message will be sent by Transmission Module indicating an unsuccessful *Start Req*. The reader may, however ask what happens when *Start Req* is successful. There is no successful *Start Req* message, so when *Start Req* is successful, Transmission Module starts sending measure messages. This message was the last to be defined in Transmission Module – Datacenter communication protocol. When in *Start Measure* mode, Transmission Module will send a measure message as soon as one oximetry measure is available, thus resulting in real-time oximetry monitoring. This operation mode will continue until a *Disconnect Req* is received from Datacenter. One measure message, besides SpO_2, Heart Rate and oximeter status information, contains time and date attached data. Detailed description on Transmission Module's measure message can be found in Annex A - Table: A.5

6.2. Bluetooth socket

In the present section will be described Bluetooth socket's Firmware Design. According to the standard design procedure, Firmware Design was made recurring to flowcharts. However, in order to reduce chapter's complexity and extension, only high level routines were considered. Detailed Flowcharts can, however, be found in Annex B.1.

This section is divided into the four major types of firmware routines which are Hardware initialization routines, Interrupt routines, Bluetooth module handling routines and main *states machine*. In order to provide a generalized overview on the Bluetooth socket's entire firmware, a general flowchart is presented in the following figure.

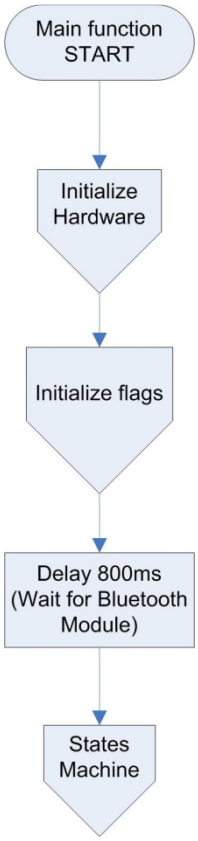


Figure 6.2: Bluetooth socket's firmware general flowchart. Note for the 800ms delay before entering the *states machine*, which is the time that Bluetooth module requires to become ready, after power up.

As it can be seen in Figure 6.2, there are several steps that the microcontroller must accomplish before entering the main program which is the *states machine*. Amongst the three major routines presented in the general flowchart only Flag Initialization will not be described due to its lower importance.

6.2.1. **Hardware Initialization**

Hardware initialization is a crucial task that must be carried out in the beginning of main program. In this initial stage, I/O ports are put in basal state and its direction is set, microcontroller peripherals are properly configured and initialized and extern modules are powered on and set in

a release state. Actions taken in the hardware initialization will thus condition the entire program flow. The following figure presents the flowchart that describes Hardware initialization step.

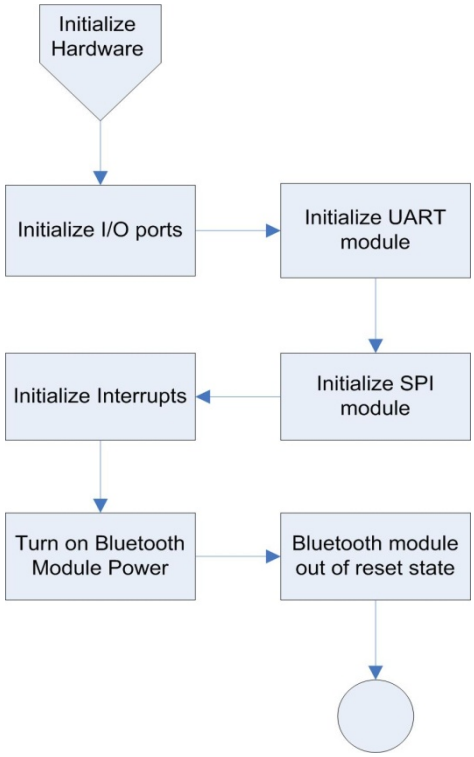


Figure 6.3: Bluetooth socket's Hardware initialization flowchart.

UART module is initialized to communicate with Bluetooth module, which results in a calculated value of 11 to BRG register (see chapter 5.3.1 – Microcontroller for BRG formula and chapter 5.3.1 – Bluetooth module for serial port configuration parameters). Along program's execution, UART module is re-initialized to communicate directly with Bluetooth oximeter, which results in a calculated value of 47 to BRG register (see chapter 5.2 - Table 5.2 for Bluetooth oximeter serial port configuration parameters).

SPI module is initialized for Slave operation and automatic CS line handling and both UART and SPI interrupts are enabled.

6.2.2. **Interrupts**

As said before, UART and SPI communication is handled by interrupt routines. Every time a UART or SPI interrupt occurs, its correspondent interrupt handling routine is executed. The following topics describe UART and SPI interrupt handling routines.

UART

The following figure presents UART interrupt handling routine resumed flowchart. It is important to make clear that UART interrupts comprise transmission (TX) or reception (RX) interrupts. Detailed UART interrupt flowcharts can be found in Annex B.1.

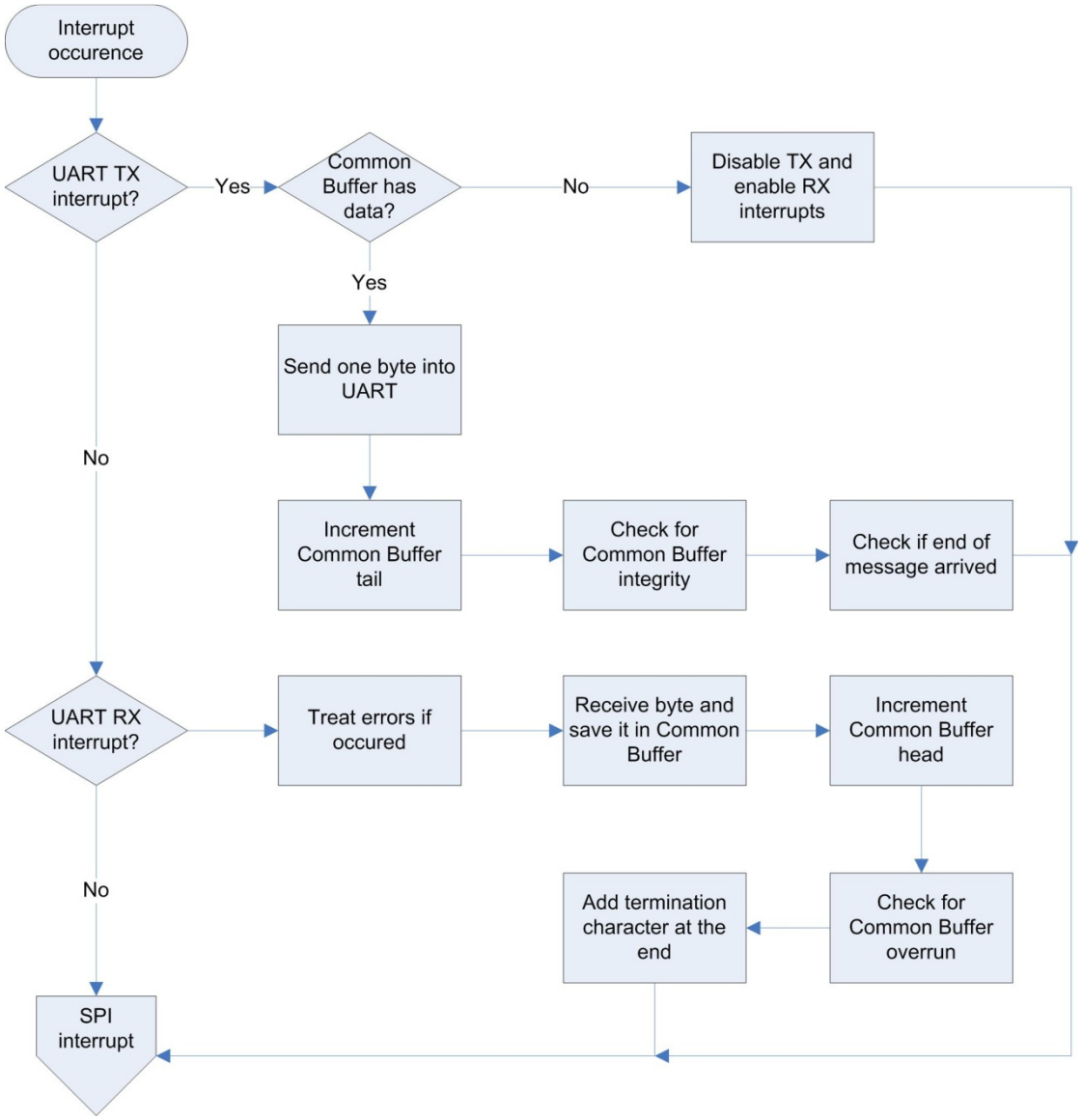


Figure 6.4: Bluetooth socket's UART interrupt handling routines' flowchart.

UART interrupts handling routines operate under a common *buffer* variable. When a TX interrupt occurs, data are continuously sent out and common *buffer*'s *tail* is continuously incremented, until it is empty. On the other hand, when RX interrupt occurs, data are saved on common *buffer* while its *head* is incremented at each received byte. *Buffer* overrun is checked each time a TX or RX UART interrupt occurs. Additionally, UART RX communication errors are properly handled if occurred.

SPI

SPI interrupt handling routine flowchart is resumed in the following figure. It is important to recall that SPI interrupt is directly related with Main Board communication (see chapter 6.1.1).

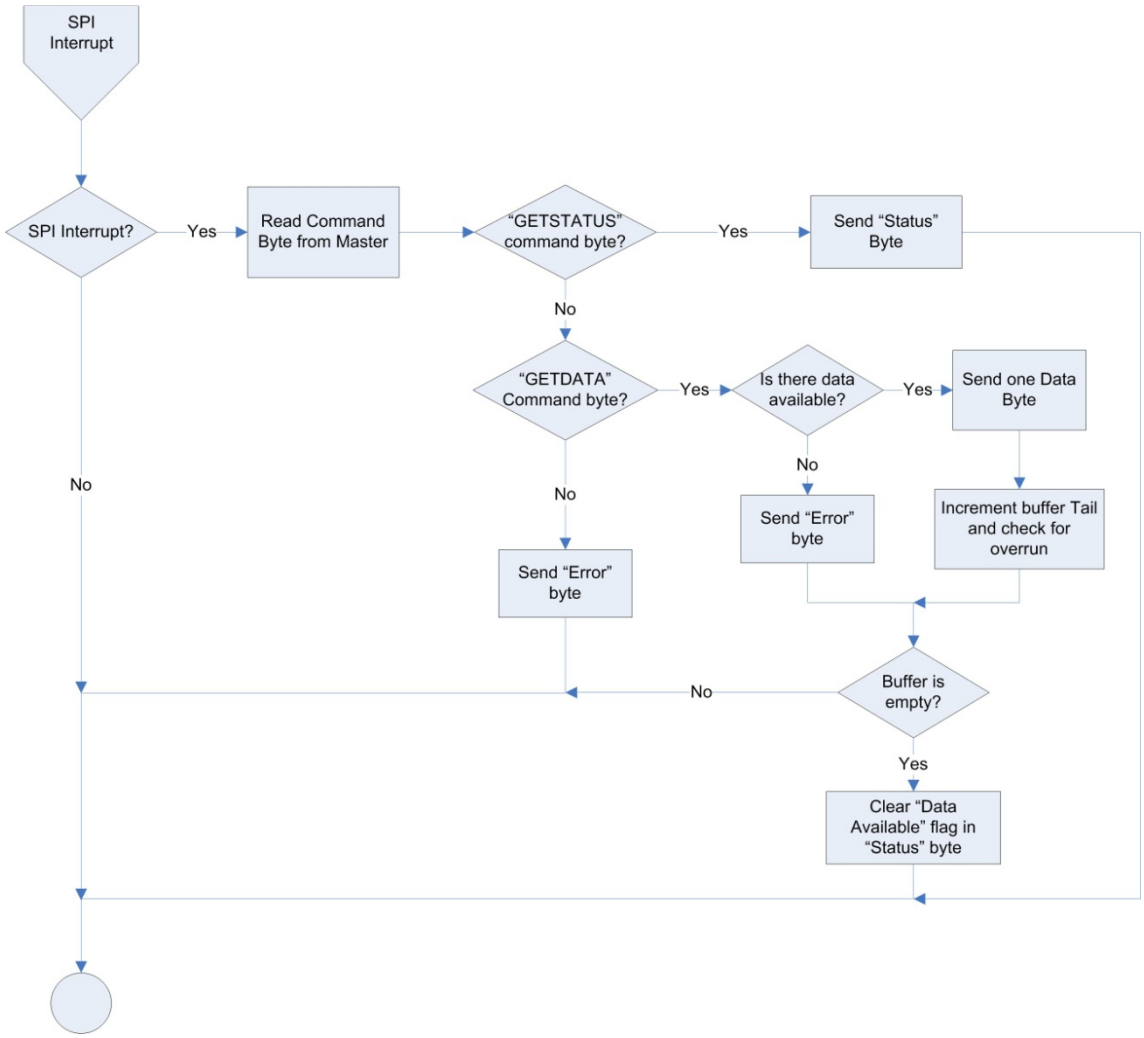


Figure 6.5: Bluetooth socket's SPI interrupt handling routine's flowchart.

Each time a SPI interrupt occurs one of three actions can occur. In case a *GETSTATUS* command byte is received, *Status* byte (see Annex A - Table: A.3) is loaded into SPI register. On the other hand, if a *GETDATA* command byte is received and there is data available, one data byte is loaded into SPI register. Otherwise, an *ERROR* byte is loaded into SPI register. Every time a command byte is received, it is checked if SPI *buffer* is empty so that *Status* byte can be updated with the new information. This allows Main Board to know that SPI *buffer* is empty the next time a *Status* byte is requested.

6.2.3. **Bluetooth Module handling routines**

In order to configure Bluetooth module with the required settings, several handling routines had to be developed, according to the module's Wireless UART protocol. Analyzing Nonin oximeter's requirements, there were identified four different configuration stages which are described in the following table.

Table 6.5: Description of Bluetooth module's configuration stages.

Configuration routine name	Description
Config BT Module	Configures Bluetooth module's Operating Mode and Serial Port
Scan for Devices	Requests Bluetooth module to start scanning for Bluetooth devices
Receive Scanned Devices	Handles the reception of found device messages from Bluetooth module
Configure Nonin Connection	Configures Bluetooth module to connect with Nonin oximeter

The following topics describe each of the identified Bluetooth module's configuration routines.

"Config BT module"

This routine configures the Bluetooth module with the desired communication parameters (see Table 6.7). In order to perform configuration, several messages are sent with a predefined sequence (see Table 6.6). A *states machine* is implemented, where each *state* corresponds to a message. After one message is sent the *state* is incremented. In the following state, before sending the next message, it is checked if the confirmation message has arrived. In case it hasn't arrived the *state* is decremented and the previous message is resent. When the last *state* is completed the *states machine* terminates its execution and the program resumes in the main *states machine*.

Table 6.6: Messages sent in the *Config BT Module* routine.

#	Message
1	*Enter HCM*
2	*Set Op Mode*
3	*Config SP*
4	*Run BT Module*

Details on each individual message sent to Bluetooth module can be found in Annex A - Table: A.6. Flowchart of the *Config BT module* routine is presented in Annex B.1.

Table 6.7: Bluetooth Module Configuration Parameters.

Parameter	Value
Operating Mode	Connecting Mode
Baud rate	9600 bauds
Flow Control	On
Parity	None
Data Bits	8
Stop Bits	1

"Scan for Devices"

This routine is responsible of putting Bluetooth module scanning for devices in the neighbourhood. The behaviour of this routine is similar to the previous one, that is, each *state* of the *states machine* corresponds to a message sent to the Bluetooth module and the next message is sent only if the previous one is confirmed. There are some parameters that need to be defined in order to properly configure the scanning process (see Table 6.9):

- A filter used when searching for devices of a certain inquiry access code (*INQ_LAP*).
- The possibility to search for a specific number of devices (*MAX_RESP*).
- The number seconds a search shall be active (*TIMEOUT*).
- A filter (if any) used when searching for devices of as certain class (*COD*).
- The possibility to include the received signal strength indication (*RSSI*) of the devices found.
- The possibility to include the remote Bluetooth name of the devices found (*RNR*).

The following table contains the sequence in which the messages are sent.

Table 6.8: Messages sent in the *Scan for Devices* routine.

#	Message
1	*Enter HCM*
2	*Scan Req*

Again, details on each individual message sent to Bluetooth module can be found in Annex A - Table: A.6. Flowchart of the *Scan for Devices* routine is presented in Annex B.1.

Table 6.9: Bluetooth Module Scanning Parameters.

Parameter	Value	Description
INQ_LAP	009E8B33 (hex)	General/unlimited inquiry access code
MAX_RESP	05 (hex)	Maximum number of devices - 5
TIMEOUT	05 (hex)	5 * (1,28) seconds
COD	00000000 (hex)	No class of device filter
RSSI	00 (hex)	No Received Signal Strength Indication
RNR	00 (hex)	No Remote Name Request

"Receive Scanned Devices"

Unlike the previously presented routines, *Receive Scanned Devices* routine is responsible not to send commands to Bluetooth module, but to handle the reception of the messages sent by that module. Typically, this function is called after the *Scan for Devices* function so that the messages returned from the Bluetooth module that contain information of Bluetooth devices found in the neighbourhood can be properly received.

This routine is continuously checking common *buffer* for new data until a timeout occurs. Anytime a message is received, its integrity will be checked and two actions can occur. In case it is a *Found Device* message (see Annex A - Table: A.6), device's MAC address will be locally stored and local *Found Devices* counter will be also incremented. On the other hand, if an *End of scan* message is received, routine will end and program's execution will resume from main *states machine*. Flowchart of the *Receive Scanned Devices* routine is, again, presented in Annex B.1.

"Configure Nonin Connection"

This routine has the purpose to configure Bluetooth module to connect with Nonin oximeter. This routine's operation is also based in a *states machine* in which each *state* corresponds to a configuration message and verification of the previous message's confirmation. Parameters of the Nonin oximeter's connection are presented in Table 6.10.

Table 6.10: Nonin oximeter's connection parameters.

Parameter	Value	Description
Bluetooth Address	(MAC ADDRESS)	Address of the Bluetooth unit that the module shall connect to
Security Mode	Mode 2	Service level enforced security
Encryption Mode	OFF	Encryption turned off
PIN Code	(PIN Code)	Pin code of the secured connection

The following table contains the sequence in which the messages are sent.

Table 6.11: Messages sent in the *Configure Nonin Connection* routine.

#	Message
1	*Set Connect Rule*
2	*Set Security Mode*
3	*Set PIN Code*
4	*Run BT Module*

Again, details on each individual message sent to Bluetooth module can be found in Annex A - Table: A.6. Flowchart of the *Configure Nonin Connection* routine is presented in Annex B.1.

6.2.4. **Main "States Machine"**

The main *states machine* is where the decisions related to the main program flow are taken. The advantage of using this kind of method is that it allows moving back and forward in the code execution, that is, it's possible to execute the states sequence as is or, for instance, to jump between non-sequential states. This advantage makes handling of unexpected errors easier. The program will repeatedly enter one *state* until it has successfully ended and then pass to the following state. If an unexpected error is detected, it's then easier to resume the program execution from the desired point.

The main *states machine* accounts for several unexpected situations, regarding both the communication with Bluetooth Module and oximeter. The *mainstate* variable increment only after the previous *state* completes. It guarantees that before executing the next step of the program the previous one is completed. The following figure presents main *states machine* flowchart.

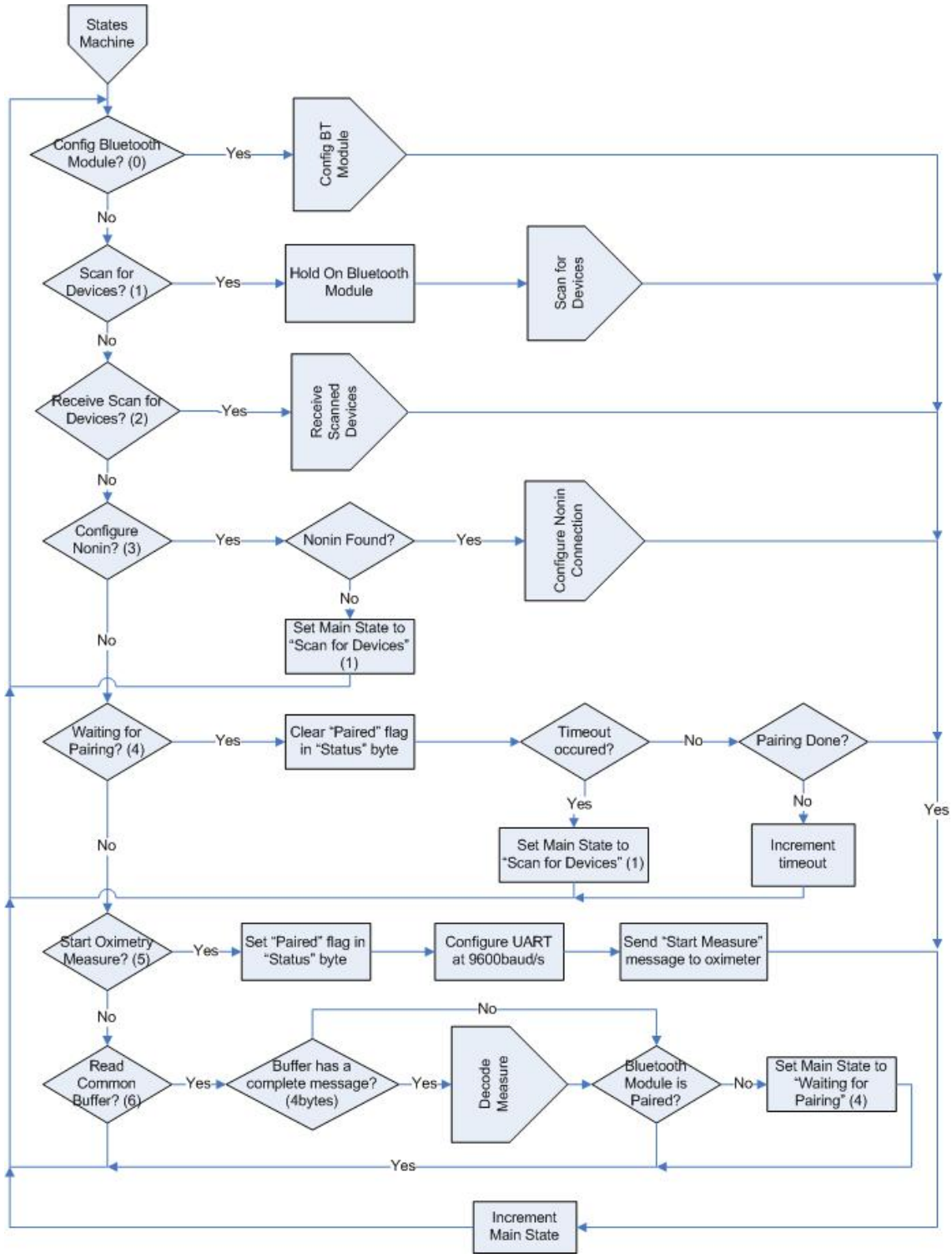


Figure 6.6: Bluetooth socket's *main states machine* flowchart.

As it can be verified by the Figure 6.6's flowchart, there are several unexpected situations that are properly handled in the main *states machine*. The first one is in case Nonin oximeter is not found after the *Scan for Devices* process. In this case, program will jump to a previous *mainstate* and re-scan for Bluetooth devices. This will occur until Nonin oximeter is found. The

second unexpected handled situation is when a *timeout* occurs while waiting for oximeter pairing process to be accomplished. Program's execution will also jump to a previous *mainstate* and re-scan for Bluetooth devices. The third and last situation regards Bluetooth unexpected unpairing while in measure process. When this situation occurs, program's execution will jump to the previous *Waiting for Pairing mainstate*.

Handling of these three unexpected situations in the main *states machine* is responsible to overcome the signal range and loss of connectivity issues of the Bluetooth Protocol.

Decode Measure routine's flowchart can be found in Annex B.1.

6.3. Ethernet Socket

Ethernet socket's Firmware Design resumed to an adaptation of the existent Microchip TCP/IP stack firmware. This section is divided into two topics. The first provides an overview report on the stack's operation while the second describes the adaptations that had to be made in order to integrate firmware into Ethernet socket's particular hardware.

6.3.1. **Microchip TCP/IP stack overview**

TCP/IP stack from Microchip is a group of programs that provide the developer services to standard TCP/IP-based applications. All the layers of the TCP/IP model are implemented by the stack's services, so that the developer just needs to mount the several firmware modules required by his solution (61).

Ethernet socket is required to work as an automatic UART to Ethernet Bridge. When connected to a LAN, Ethernet socket will automatically obtain IP address by DHCP. As soon as it is registered in the network, it will try to establish a connection as a TCP client to a previously defined IP address and port. When TCP connection is established a UART to Ethernet Bridge is created and information can be transparently exchanged between Main Board's UART connected microcontroller and Ethernet connected Datacenter (61).

6.3.2. **Adaptations**

Microchip's TCP/IP stack was designed to work with a specific hardware board, so some adaptations had to be made in to order to integrate the firmware into the existent Ethernet

socket's hardware. Adaptations resumed to three main areas which include I/O ports remapping, Oscillator configuration and UART baud rate setting.

I/O ports remapping

I/O ports remapping adaptation resumed to modify stack's Hardware Profile in order to remap SPI, LED and UART pins as well as setting their direction. Some additional adaptations had to be made regarding some existent button pressing events' routines.

Oscillator configuration

Oscillator configuration was made taking into account PIC24HJ's PLL system (see chapter 5.3.2 - Microcontroller).

N_1, M and N_2 values were calculated in a way that microcontroller could operate at maximum possible speed. Taking into account frequency limitations at the exit of prescaler, voltage controlled oscillator and postscaler, the maximum allowed FOSC was calculated in 79.2576MHz for a 14.7456MHz oscillator (see Table 5.5). N_1, M and N_2 values are then 4, 43 and 2, respectively and F_{CY} is half the FOSC value or 39.6288MHz.

UART baud rate configuration

UART baud rate communication between Main Board's and Ethernet socket's microcontrollers was defined to be 57600 baud which is more than enough considering the amount of data transferred through Ethernet socket. According to BRG formula, (see chapter 5.3.1 - Microcontroller) calculated value is 42.

6.4. Main Board

Main Board's Firmware Design is presented in this section. Again, only high level routines will be presented, while detailed flowchart can be found in Annex B.1. Main Board's microcontroller is the responsible for taking the main decisions that are related with Transmission Module. Bluetooth and Ethernet sockets are also controlled by this microcontroller.

This section is divided in four main parts. The first is related with data storage organization within EEPROM and Flash memories and the following three describe the major types of Main Board's firmware routines. These include Hardware initialization routines, Interrupt

routines and main *states machine*. A generalized overview is provided in the following figure by means of the general Main Board's microcontroller flowchart.

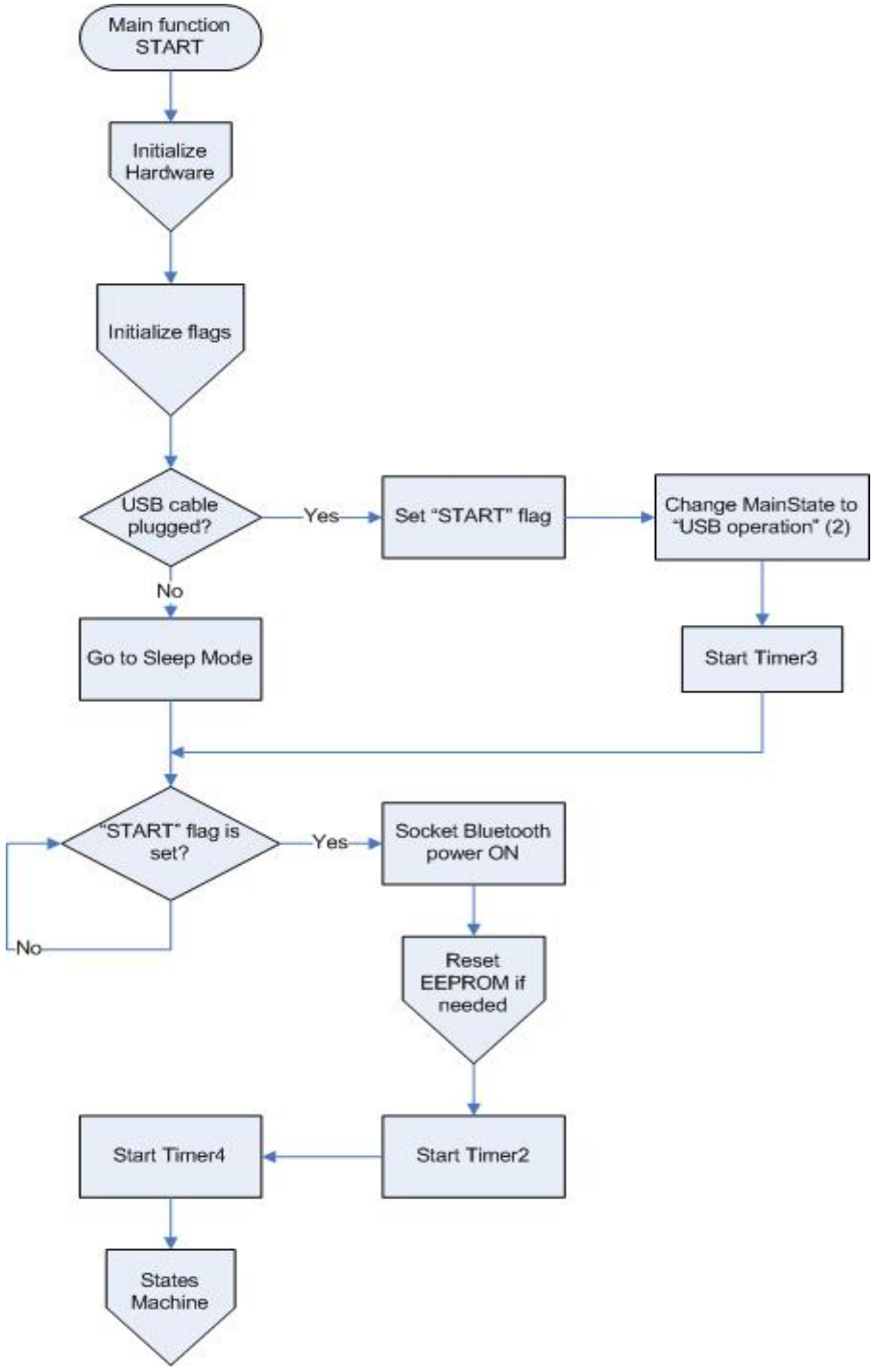


Figure 6.7: Main Board's firmware general flowchart.

As it can be seen in Figure 6.7, after initialization routines a decision is taken. In case there is a USB cable plugged-in, execution continues until it reaches *state 2* of the *states machine*. Otherwise, microcontroller will enter *sleep* mode until an interrupt occurs and only then

program's execution will continue. *Start* flag is responsible for unleashing program's execution, in other words, until *Start* flag is set program execution will be in an infinite cycle, as it can be seen in general flowchart.

It is important to make clear that Main Board's operation is based on two main timer routines. *Timer 2* handles Bluetooth socket data retrieving and storage in Flash memory and *Timer 3* handles data retrieving from Flash memory into a local variable. *Timer 4*, on the other hand, refreshes local time structure data.

Amongst the major routines of the general flowchart, *Initialize Flags* and *Reset EEPROM* will not be described but can be found in Annex B.1.

6.4.1. **Data Storage architecture**

As it was previously said (see chapter 5.3.3 – Memories), measure's data storage is made through both EEPROM and Flash memories. EEPROM memory has a configuration purpose while Flash memory has a data storage purpose.

This dual purpose architecture allows Flash memory *head* and *tail* pointers to be stored in EEPROM thus allowing data to be maintained when power is lost and program unexpectedly terminates. EEPROM memory organization is presented in the following table. It is important to recall that Flash memory data pointers are constituted by *Page* and *Byte* values (see chapter 5.3.3 – Memories).

Table 6.12: EEPROM's configuration memory internal organization.

Byte Number	Address Type	Pointer Type
0	*Page*	*Head*
1		
2	*Byte*	
3		
4	*Page*	*Tail*
5		
6	*Byte*	
7		

In order to allow the microcontroller to check for memory integrity, a verification byte was also included in EEPROM byte position 8. In case this byte has the value *AA* hexadecimal

EEPROM memory is properly formatted, otherwise memory is corrupted and a reset must be made. EEPROM memory reset consists on erasing the values retained in byte positions from 0 to 7 and load the value *AA* hexadecimal into position 8.

Following this data storage architecture, every time one measure is saved in Flash memory, its *head* and *tail* pointers will be refreshed and saved in EEPROM memory in the proper positions. On the other hand, when, somewhere in program's execution Flash memory's data occupation must be known, it is sufficient to access EEPROM's stored *head* and *tail* pointers.

6.4.2. **Hardware Initialization**

Main Board's hardware initialization divides into Oscillator configuration, I/O ports initialization and peripherals configuration. This step of the program is executed in the beginning of main program and constitutes, as it was previously said, a crucial task since it will condition the remaining program execution. Hardware initialization flowchart will be presented in the following figure.

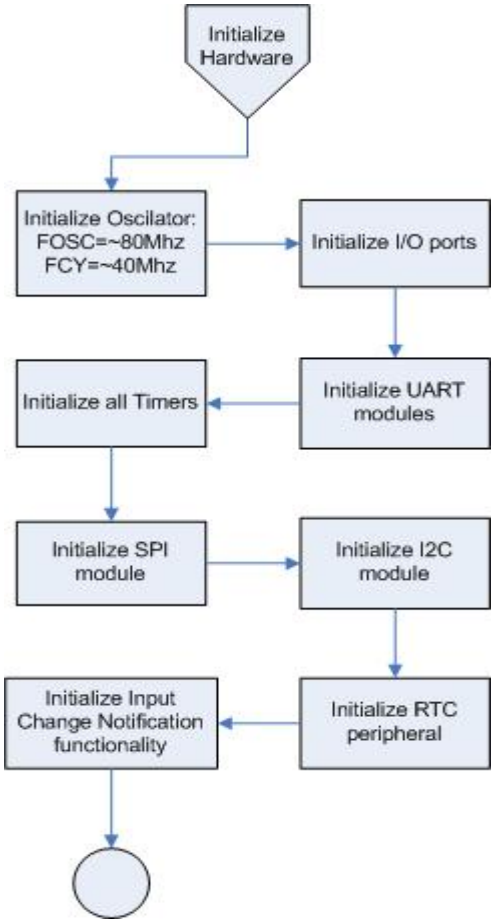


Figure 6.8: Main Board's Hardware initialization flowchart

Oscillator configuration is made in the same way than in Ethernet socket's microcontroller. FOSC value is 79.2576MHz which corresponds to N_1, M and N_2 values of 4, 43 and 2, respectively and F_{CY} value is 39.6288MHz.

UART module is initialized to communicate with both Ethernet socket and USB transceiver with a baud rate of 57600 baud (see chapter 6.3.2 - UART baud rate configuration). This way, calculated BRG value is 42, which correspond to Ethernet socket's value.

SPI module is initialized to work as a Master device and all CS lines that are controlled by Main Board's microcontroller are put at its idle state. Furthermore, clock line is configured to its lower operation frequency in order to minimize data loss, since transferred data amounts are low.

I_2C module is also initialized in Master mode with a clock operating frequency of 396.435KHz. RTC unit's configuration registers are also initialized.

Timers, UART module and Change Notification interrupts are enabled throughout Hardware initialization routine.

6.4.3. **Interrupts**

As said before, UART, Timers and Change Notification are handled by interrupt routines. The following topics describe the routines that handle each one of the presented interrupt types.

UART

Every time one byte is received in UART module an interrupt is generated and its correspondent routine will be executed. Figure 6.9 presents the flowchart of UART reception (RX) handling routine.

UART RX interrupt handling routine operates under a dedicated *buffer* variable. When a RX UART interrupt occurs, data are saved on RX *buffer* while its *head* is incremented at each received byte. *Buffer* overrun and collision errors are properly identified and treated. After received data are saved in RX *buffer*, UART RX interrupt routine will check if any message request was received from Datacenter. There are three possible request messages, as presented in chapter 6.1.2 - Table 6.3. In case a *Connect Req* message is received a *Connect Resp* will be sent to Datacenter. On the other hand, if a *Start Req* message is received two actions can occur: if Transmission Module is connected, program execution will enter *Send Data* mode, otherwise a *Start Resp* error message will be sent to Datacenter. Finally, in case a *Disconnect Req* message is received a *Disconnect Resp* message will be sent to Datacenter. Detailed description of *Send Connect Resp* and *Send Disconnect Resp* subroutines can be found in Annex B.1.

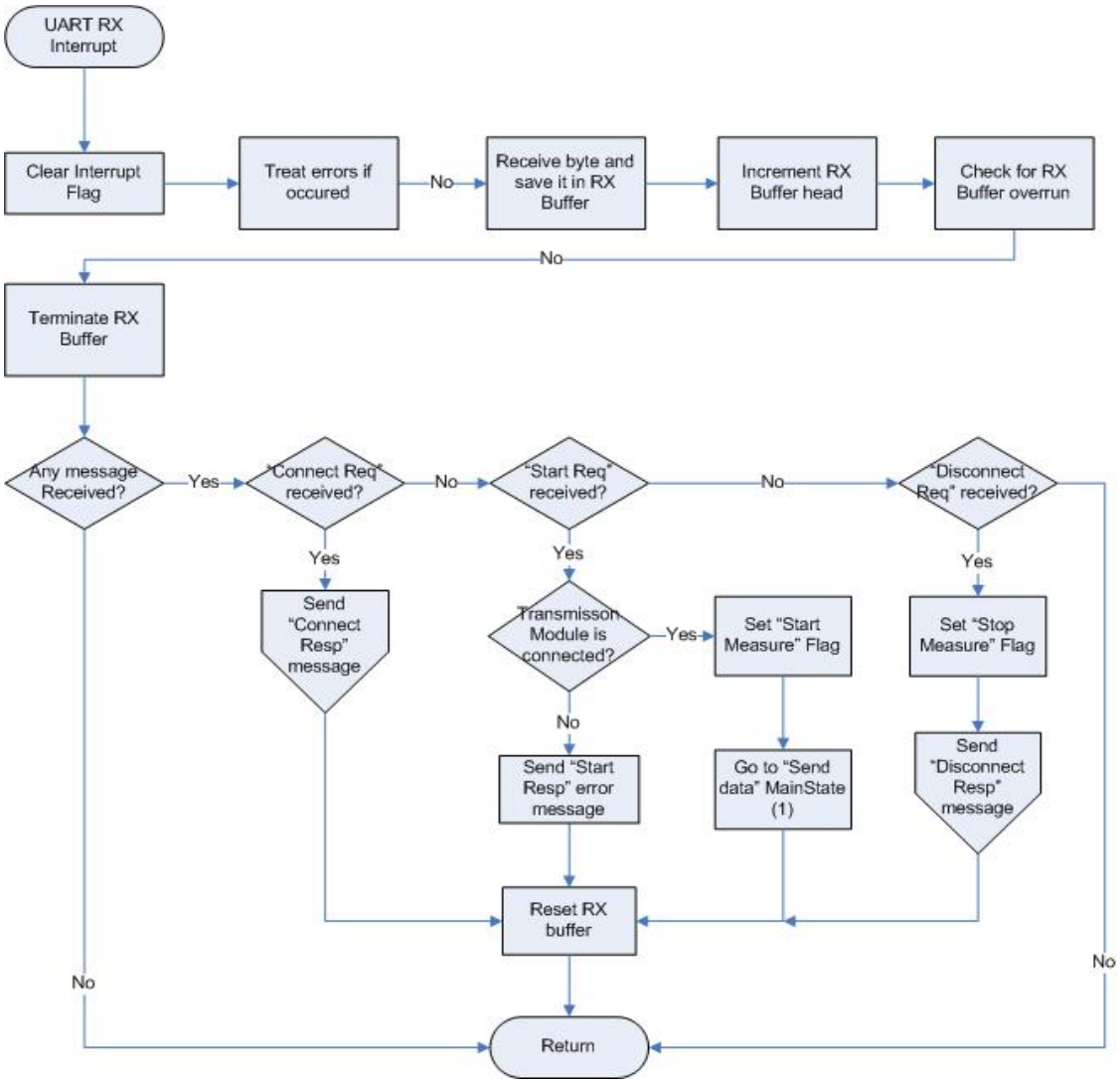


Figure 6.9: Main Board's UART RX interrupt handling routine's flowchart.

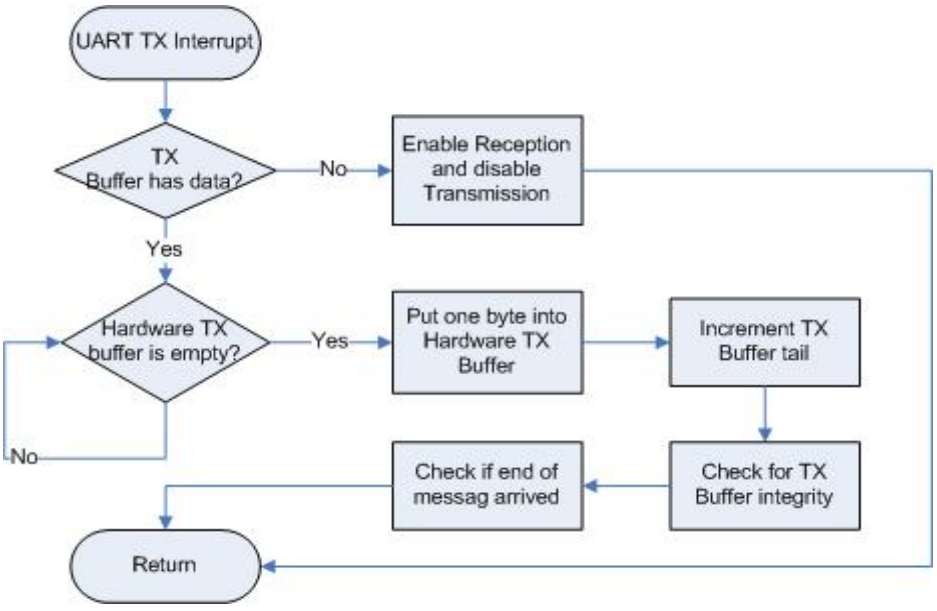


Figure 6.10: Main Board's resumed UART TX interrupt handling routine's flowchart.

Figure 6.10 presents the resumed flowchart of UART transmission (TX) handling routine (see Annex B.1 for detailed flowchart). UART TX interrupt handling routine also operates under a dedicated *buffer* variable. When TX interrupt is forced by software, data are continuously sent out and TX *buffer*'s *tail* is continuously incremented, until it is empty. Several verifications are made in the UART TX interrupt handling routine. Routine will wait until Hardware TX *buffer* is empty before sending the data byte. Additionally, software TX *buffer* overrun and end of message arrival is checked.

Timers

Despite five timer units are used throughout entire firmware program's execution, only *Timer 2* and *Timer 3* handling routines will be described which execute, as said, the most important tasks. Additionally, *Timer 4, Timer 6 and Timer 8* handling routines are responsible for local time structure data refreshing from RTC unit, user interface button and USB cable plug/unplug handling events, respectively and can be found in Annex B.1.

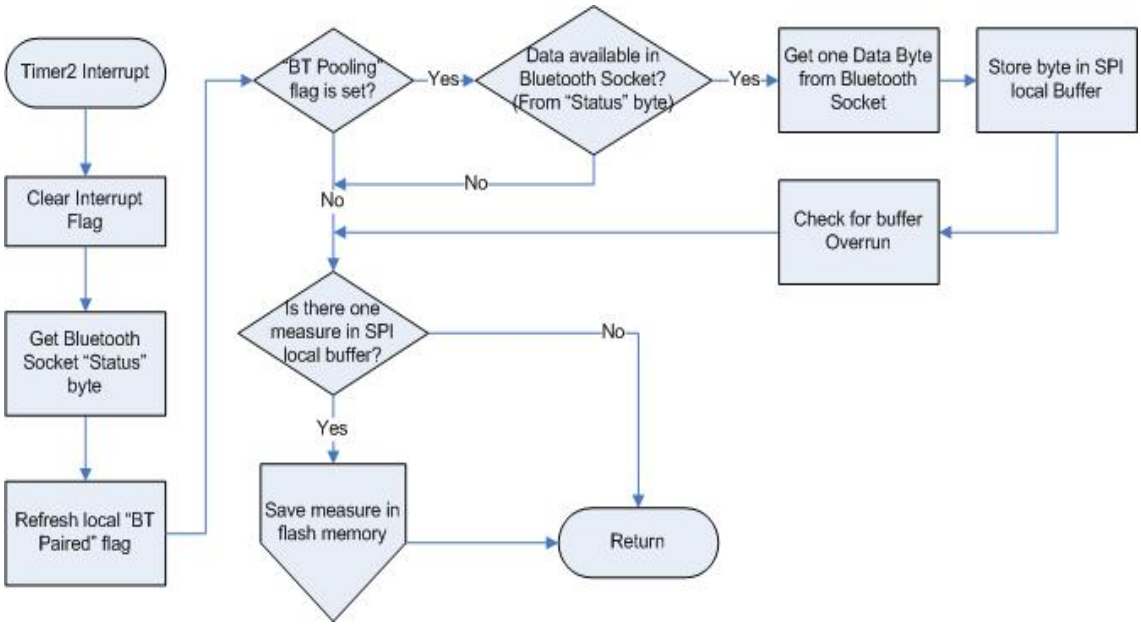


Figure 6.11: Main Board's resumed *Timer 2* interrupt handling routine's flowchart.

As said before, *Timer 2* interrupt handling routine has two distinct tasks which comprise the data retrieval from Bluetooth socket and correspondent storing in Flash memory. A resumed flowchart of *Timer 2* interrupt handling routine is presented on Figure 6.11, but a detailed flowchart can be found in Annex B.1.

Timer 2 interrupt handling routine is executed with a periodicity of 100ms, in other words, Bluetooth socket is pooled every 100ms which is considered a good compromise between acquired data latency and processor consumed resources. First, Bluetooth socket *Status* byte is retrieved in order to refresh the local flag that indicates whether the oximeter is paired or unpaired. Following this initial task, two actions can occur: in case Bluetooth pooling is turned on, one data byte will be retrieved from Bluetooth socket and stored locally; and if there is one entire oximeter measure stored, which comprises four bytes (see chapter 5.2 - Table 5.1), time and date information will be attached and it will be stored in Flash memory. *Save Measure in Flash memory* routine is presented in Annex B.1.

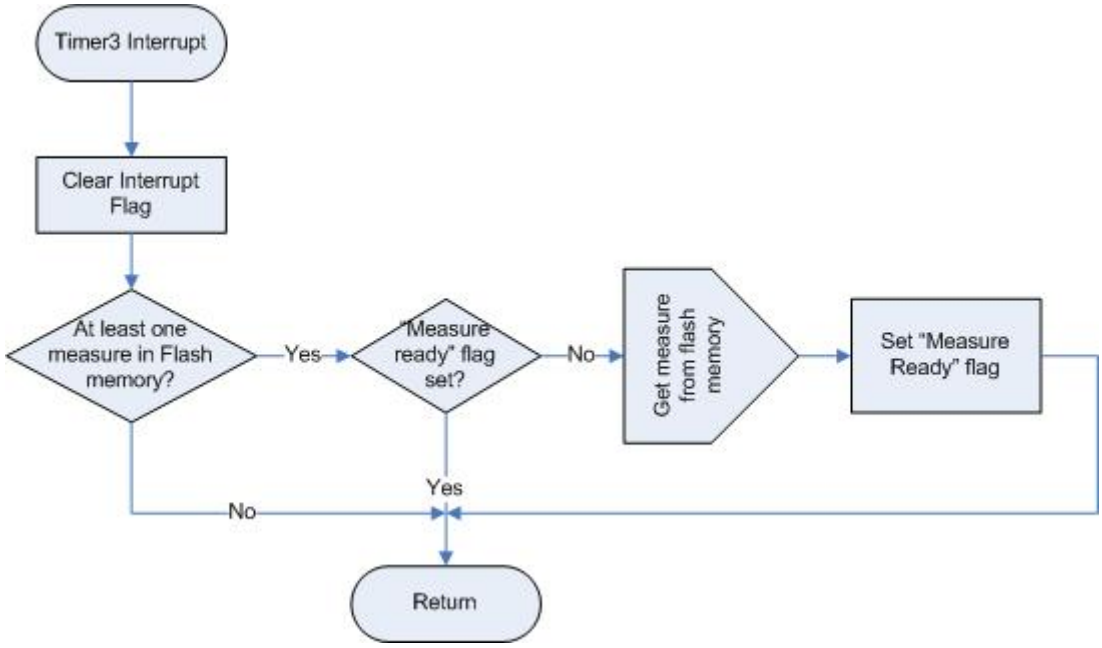


Figure 6.12: Main Board's *Timer 3* interrupt handling routine's flowchart.

The maximum delay from the instant that a measure is available in Flash memory and it is sent to Datacenter is 200ms which is the periodicity of *Timer 3* interrupt handling routine (see Figure 6.12). Firstly, it is important to refer that *Measure Ready* flag, when set, indicates the main program when a measure is ready to be sent to Datacenter and is cleared after the measure is sent. When the routine is executed, in case there is at least one measure stored in Flash memory and *Measure Ready* flag is already cleared, one measure is retrieved from Flash memory and stored locally. *Measure Ready* flag is finally set in order to indicate the main program that the measure is ready to be sent. *Get Measure from Flash memory* subroutine can also be found in Annex B.1.

Input Change Notification

Input Change Notification interrupts concerns both user interface button and USB cable plug/unplug events. Input Change Notification module of the microcontroller generates an interrupt every time a pin state change occurs. Only user interface button and USB cable plug/unplug indication pins have Input Change Notification enabled, so when an interrupt occurs, its correspondent handling routine must find out which of these two pins changed its state. This is done by storing the previous state of each pin so that when the interrupt occurs the routine can compare the actual pin state to the previous one. The following figure presents the flowchart of the Input Change Notification handling routine.

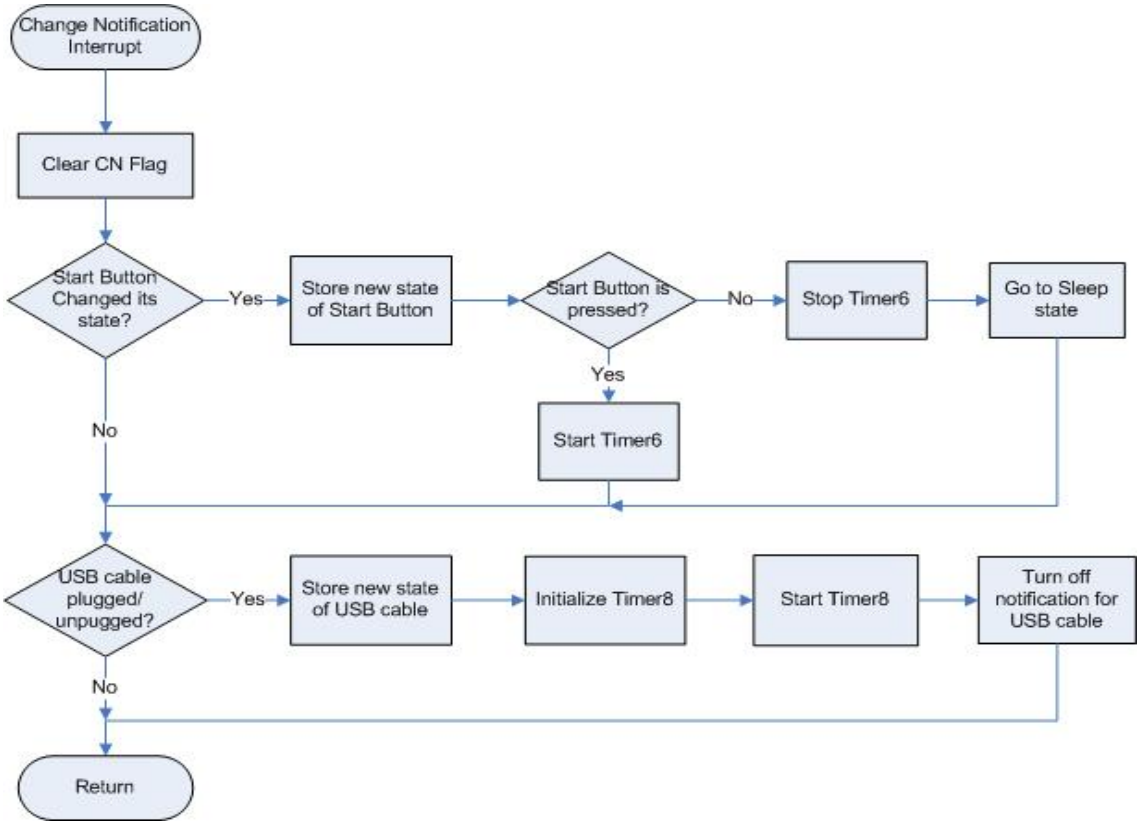


Figure 6.13: Main Board's Input Change Notification interrupt handling routine's flowchart. Start Button is the same as user interface button.

According to the pin whose state changed, a different action will be taken. *Timer 6* and *Timer 8* will be activated in case user interface button or USB cable events occur, respectively. This way, the remaining event handling will be done by *Timer 6* and *Timer 8* interrupt handling routines (see Annex B.1).

6.4.4. **Main "States Machine"**

Main Board's main *states machine* is composed only by three states which indicates that Main Board's firmware program divides processor power mainly by interrupt routines. This mode to operate has the advantage of reducing time delays for tasks accomplishment.

Main *states machine's* flowchart that is presented in the following figure, shows that each of three states is responsible for one entire operation mode.

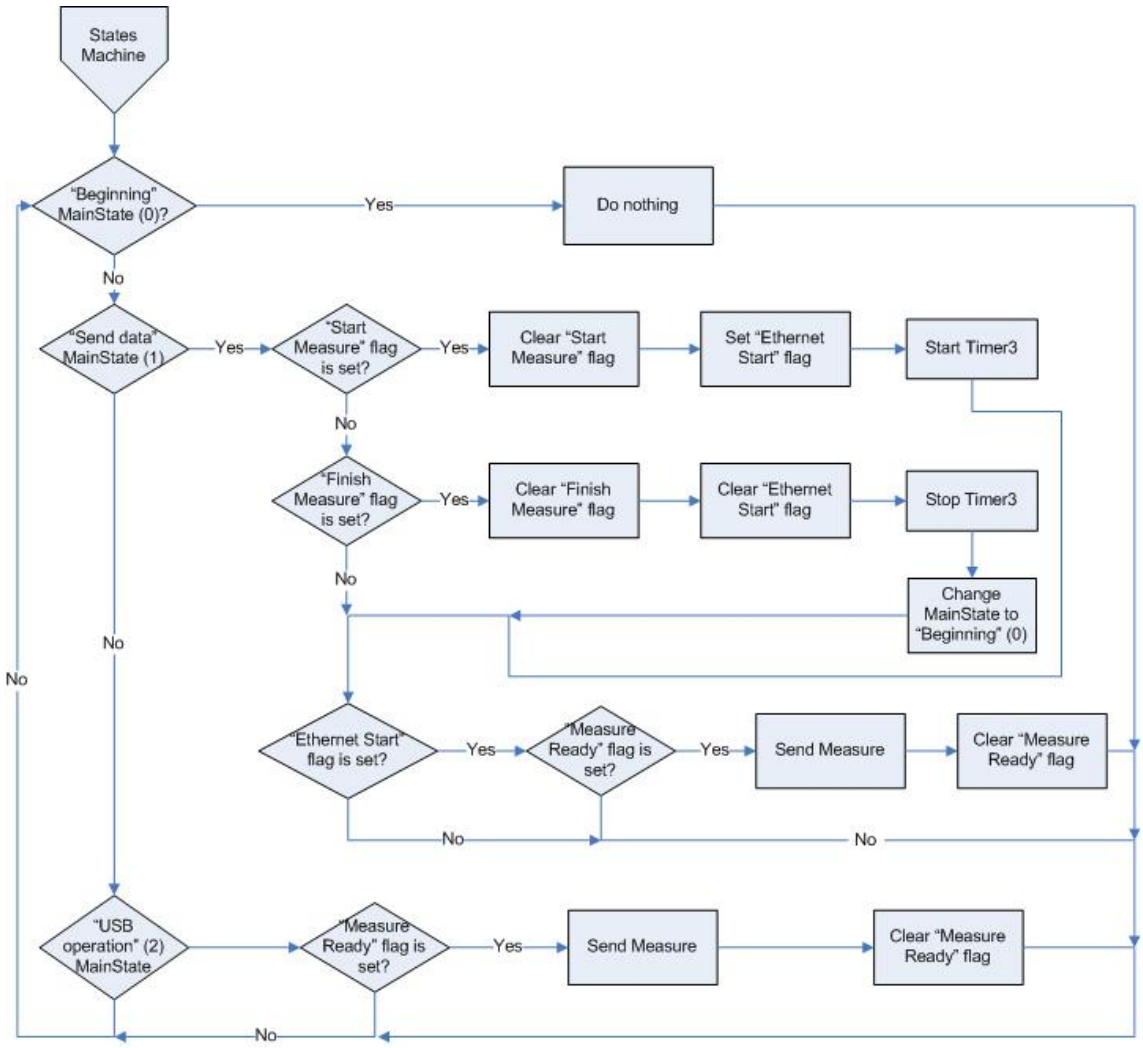


Figure 6.14: Main Board's main *states machine* flowchart.

The first *state* (0) which is called *Beginning mainstate* is selected when Module's program execution is on hold. This *state* is often executed while Transmission Module is retrieving oximeter's data and storing it internally without sending it through Ethernet or USB.

On the other hand, *Send Data mainstate* is responsible for handling program's execution while Transmission Module is retrieving oximeter's data and simultaneously transmitting it to

Datacenter through Ethernet. Inside this *state* three actions can be taken. In the first two, *start* and *finish* measure tasks will be accomplished while the last one is responsible for sending measure to Ethernet socket in case it is ready to send. It is important to recall that *Measure Ready* flag is also handled in *Timer 3* interrupt routine and *Start* and *Finish Measure* flags are handled in UART RX interrupt routine, that were previously presented.

Finally, *USB Operation mainstate* is executed while sending measure data through USB port. When in USB operation, there are not any *start* or *finish* measure messages, so only one action can be taken which consists on the sending of measure data to USB port, in case it is ready to be sent.

Chapter 7

7. Tests and Final Results

Considering the state of conclusion of the project it was considered important to include this chapter, where the result of some tests carried out in the end of the project's course are presented. The chapter is divided into two sections which comprise Hardware and Firmware Operation tests, only related to Transmission Module.

7.1. Hardware Tests

Transmission Module hardware tests were performed in collaboration with Tomé Matos, however only the tests that are somehow related with firmware are herein presented.

Besides the regular hardware tests, several firmware routines were made in order to inspect several individual components of the Transmission Module, including Main Board, Bluetooth socket and Ethernet socket. The following tables indicate each of the tested component, its validation and a short description on how the test was made.

Table 7.1: Individual Main Board components tests' description.

Component	Validation	Description
LED's D5, D6, D7 and D8	OK	PIC pin state alternation and visual verification.
Buttons S4 and S5	OK	Cyclical reading of the pin state value and LED lighting in case it is pressed
RS-485	OK	Communication with a PC through a RS-232 converter
SPI (EEPROM)	OK	EEPROM numerical sequence writing and reading and (RS-485 result printing)
SPI (Flashes)	OK	Flashes numerical sequence writing and reading and (RS-485 result printing)
I_2C (RTC)	OK	RTC unit configuration and periodical time and date retrieving (RS-485 result printing)
USB	OK	Communication with a PC's USB port.

Table 7.2: Individual Bluetooth socket components tests' description.

Component	Validation	Description
LED's D1, D2	OK	PIC pin state alternation and visual verification.
SPI (Main Board)	OK	Send and receive a predefined byte to and from Main Board

Table 7.3: Individual Ethernet socket components tests' description.

Component	Validation	Description
LED D4	OK	Microchip stack operation.
SPI (EEPROM)	OK	Microchip stack operation.
SPI (Ethernet Controller)	OK	Microchip stack operation.

Some of the routines elaborated to accomplish the referred tests were an important starting point to the low-level routines that integrated the final firmware program.

7.2. Firmware Operation Tests

Like in the previous section, some of the Transmission Module firmware operation tests were made in collaboration with Tomé Matos, more specifically Bluetooth communication related tests. This section is then divided into three main topics which are Bluetooth communication, Main Board internal tasks and Ethernet communication tests.

7.2.1. **Bluetooth Communication**

In order to fully test Bluetooth communication between Transmission Module and oximeter, several situations were intentionally created. These include low range operation, operation through one, two and three walls and loss of range situations.

While in low range operation, the system performed flawlessly as expected and measure periodicity was never lost. On the other hand, while operating through one, two or three walls it were detected very small latency periods face to normal operation, however any measure was lost. Table 7.4 presents the maximum communication range verified in each of the created situations.

Pairing and Unpairing times were also calculated. The time elapsed between the loss of Bluetooth range and the instant the module effectively unpairs is 20 seconds while pairing time is just 4 seconds.

Table 7.4: Bluetooth maximum range through several types of barriers

Barrier	Maximum Range (meters)
Line of Sight	80
One wall	50
Two walls	22
Three walls	15

7.2.2. **Main Board Internal Tasks**

Main Board's internal tasks tests concern both proper memory data storage and correct handling of the entire firmware program's flow.

In order to verify the referred functions a continuous monitoring operation was simulated during three hours. It was verified that during the test time entire Transmission Module performed as requested and did not presented any error situation. Measure's data was properly stored and remotely transmitted in real-time.

Battery duration was also studied during this simulation. The maximum duration of the battery was two hours and ten minutes.

Concerning memory maximum data storage, some calculations were made which indicated the capacity of storing 196608 measures (each measure has 11 bytes – see chapter 5.3.3 – Memories for Flash memory capacity). Considering an average of one measure per second, the memory allows storing oximetry data during 54.61 hours.

7.2.3. **Ethernet Communication**

Ethernet communication tests were made recurring to small LAN with only one PC and one Transmission Module. Using a specific software program a TCP/IP connection was established between the PC and the Transmission Module. Request messages were then sent from the PC to Transmission Module, simulating Datacenter's operation and measure messages were properly received and analyzed in the PC. This procedure was repeated several times and Transmission Module performed as requested and any error situation was detected.

Chapter 8

8. Conclusion

In the present thesis were presented the several stages of the project carried out during the past academic year that lead to the development of a Vital Signs Remote Monitoring system. In order to achieve the final objective, several intermediate steps had to be accomplished. These include knowledge acquisition, project requirements analysis, study of system specifications and finally the firmware design which combines all the previous accomplished steps.

This document is concluded with a very brief overview on the project's plan in order to recall the progress made. Remote Vital Signs Monitoring project status is also discussed in this chapter. Furthermore, some future work considerations will be provided in order to provide future project developers a solid starting point. Finally, a summary will be presented, providing a personal appreciation on the project's benefits for general society and also its future potential. Moreover, considerations regarding the project's influence on the student's personal development will also be made. Lastly and combining all the considerations made throughout this chapter it will be possible to conclude over the project's objectives achievement.

8.1. Project Status

The main goal of this project was to accomplish the development of a system that monitors patient's vital signs, while providing high mobility and constant remote access by clinicians.

Despite being a prolongation of the previous year project, Remote Vital Signs Monitoring suffered some planning modifications which led to the abandonment of the work carried out by previous year project students. The new planning replaced the acquisition module by individual market available wireless sensors which introduced an integration component into the project. Data transferring into a remote Datacenter maintained the same architecture. Project's objective was, then, changed to the development of a Transmission Module that integrates existent

Bluetooth enabled vital signs sensors and has the capability to remotely communicate with a Datacenter in order to be included in both hospital and Telehomecare environment. Along project's execution a personalized solution was requested which comprised the integration of a single vital sign and Ethernet as the wide range remote vital signs transferring technology.

In order to carry out the selected objectives, a project requirements analysis allowed the segmentation of the several system needs according to each of its separate functionality. Vital signs acquisition, short and wide range communication and Transmission Module individual capabilities were analyzed. Despite the requirement analysis approached all the major project objectives, the development stage resumed only to the personalized and more minimal solution.

Regarding only the personalized solution, system specifications were completely indentified which include system architecture, sensor, Transmission Module and Datacenter detailed features.

Once all the needed tools were prepared, actual system development stage started, which consists on firmware development. At this final stage, all the intermediate tasks carried out along the past year were linked together and put in practice which resulted in the elaboration of a system prototype that performs all the tasks that were required. More specifically, the developed prototype integrates a wireless Bluetooth oximeter device, a Transmission Module and a Datacenter. Tests carried out in the end of project's course indicated that the developed solution performed all the required tasks of the personalized requested vital signs solution. The student's individual contribution for the project's completion is mainly related to Transmission Module's firmware development, more specifically, sensor and Datacenter integration.

However, the planning proposed for this project was not entirely concluded. The integration of ECG, accelerometer and temperature sensors was not accomplished and wide range technologies like Wi-Fi or GSM/GPRS were not included in the developed system, despite some steps were taken in order to accomplish it. The lack of time is the main reason that prevented the planning's full conclusion. Most of the project time was spent, not in system development, but in the acquisition of the knowledge required to do it. Besides that, some hardware limitations conditioned more than one Bluetooth sensor's integration.

In spite the current Remote Vital Signs Monitoring system is at a conclusion stage and properly performs its required functions, there are several features that it does not include and may be object of future work.

8.2. Future Work

This section's purpose is to provide future development suggestions so that they can be analyzed, tested and possibly implemented.

Firstly and at a short term the planning made for this year's project should be concluded. The developed system provides high integration possibilities, allowing the inclusion of both Wi-Fi and GSM/GPRS solutions. However, a hardware limitation must be surpassed in order to allow integration of more than one Bluetooth wireless sensor. One possibility is to study the use of Zigbee as the low-range communication technology, taking advantage of some research made during the present project.

Secondly, besides the planned ECG, accelerometer and Temperature vital signs inclusion, new vital signs' inclusion could be studied with the intent of providing a valuable clinical contribute. Clinical trials should always be kept in mind because they provide an important indication of the system's suitability. Still regarding sensors, miniaturization and increasing portability should be an important aspect to improve in order to provide the best comfort to the patient. System's high degree of personalization should be a feature always present and be continuously improved in order to allow integration with several other sensors or operation environments.

Finally and at long term, while in Telehomecare environment, system's integration in an Ambient Assisted Living concept should be studied. Allying the strengths of both concepts it could lead to the opening of wider but equally noble application areas.

8.3. Summary

The final summary of this document comes upon the project's final appreciation. From the personal point of view, the project concept allows a more effective and earlier diagnosis which allows both sickness prevention and a safer and more comfortable vital signs monitoring. The system allows clinicians to access real-time vital sign information in a remote location which results in a more accurate attendance and continuous monitoring. Real time alarms are also provided and constitute an important feature in providing shorter response delays.

Regarding personal benefits, the student's appreciation is very positive. In first place the participation in an innovative project introduced a big amount of motivation and the feeling of making something very useful. The opportunity to acquire new knowledge was one of the most

important project's personal achievements which join to the fact that the field of electronics and firmware programming is highly appreciated by the student. Furthermore, the participation in an enterprise environment provided not only a valuable help in project's development but also a good contribute to learn how to work among a big team and to get a good perspective of the professional world. Due to the fact that the project is composed by several elements, it provided the student a good sense of team work and how to join efforts into the same common goal. The many and differentiated difficulties that were encountered during project's execution and were surpassed with a high amount of dedication and extern help provided the student an increased problem solving capability.

The evolution of the student's both soft and hard skills from the beginning of the project leaves the sensation of task accomplished and provides more confidence to face the professional world

.

REFERENCES

1. *Estimativas de População Residente, Portugal, NUTS II, NUTS III e Municípios.* **Instituto Nacional de Estatística.** s.l. : INE, 2007.

2. *Telehomecare and Remote Monitoring: An Outcomes Overview.* **Stachura, Max E. and Khasanshina, Elena V.** Augusta : Medical College of Georgia.

3. **Shariq, Katherine.** *Remote Monitoring of Pulse Oximetry - Improving Patient Care.* [Online] 12 2004, 19. [Cited: 08 5, 2008.] http://www.frost.com/prod/servlet/market-insight-top.pag?docid=29726668.

4. *Vital Sign Monitoring in Nursing Home for eldery.* **Legarreta, J. H. and Paloc, C.** Luxemburg : Telehealth, Telecare and Services for the Ageing, 2008.

5. **Gardner, Nigel.** *An Introduction to programming the Microchip PIC in CCS C.* s.l. : Bluebird Electronics, 2002.

6. **Kernighan, Brian W. and Ritchie, Dennis M.** *The ANSI C Programming Language.* Second Edition. s.l. : Prentice Hall, 1988.

7. **Cox, Taylor.** *Migrating from Assembly to C for 8-bit Microcontrollers.* San Francisco, California : Atmel Corporation, 2004.

8. *Sensors, Microcontrollers and Ubiquitous Computing with a Wearable Computing Intermission.* **Kranz, Matthias.** Munich : University of Munich, 2005.

9. *PIC18FXXX DFT Hands On Workshop.* s.l. : Microchip Master's, 2002.

10. **Matic, Nebojsa.** PIC microcontrollers. *mikroElektronika.* [Online] 2003. [Cited: 08 10, 2008.] http://www.mikroe.com/en/books/picbook/0_Uvod.htm.

11. **Verle, Milan.** PIC Microcontrollers. [Online] 2008. [Cited: 08 11, 2008.] http://www.mikroe.com/en/books/picmcubook/.

12. *MPLAB IDE User's Guide.* **Microchip.** 2006.

13. **HI-TECH.** PICC 18. [Online] [Cited: 08 11, 2008.] http://microchip.htsoft.com/products/compilers/picc18procompiler.php.

14. **Microchip.** MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs. [Online] [Cited: 08 11, 2008.] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010065.

15. *MPLAB ICD2 User's Guide.* **Microchip.** 2007.

16. **Labcenter Electronics.** The VSM Advantage. [Online] [Cited: 08 12, 2008.] http://www.labcenter.co.uk/products/vsm_overview.cfm.

17. **Bray, Jennifer and Sturman, Charles F.** *Bluetooth 1.1 Connect Without Cables.* s.l. : Prentice Hall, 1999.

18. **Bluetooth SIG.** How Bluetooth Technology Works. [Online] [Cited: 08 13, 2008.] http://www.bluetooth.com/Bluetooth/Technology/Works/.

19. **BlueTomorrow.** The Compreensive Guide to Everything Bluetooth Related. [Online] [Cited: 08 13, 2008.] http://www.bluetomorrow.com.

20. **Free2Move.** Free2move Bluetooth Module - F2M03GLA. [Online] [Cited: 08 13, 2008.] http://www.free2move.se/index.php?type=view&pk=24&pk_language=1.

21. *F2M03GLA Datasheet.* **Free2Move.** 2006.

22. *Wireless UART firmware version 4.* **Free2Move.** 2006.

23. **Nonin Medical.** 4100 Bluetooth Oximeter. [Online] [Cited: 08 14, 2008.] http://www.nonin.com/products.asp?ID=29&sec=2&sub=9.

24. *Wireless 4100 Digital Pulse Oximeter with Bluetooth.* **Nonin Medical.**

25. *Nonin Model 4100 containing Bluetooth Technology Specification.* **Nonin Medical.** 2006.

26. **Hill, E. and Stoneham.** Practical applications of pulse oximetry. [Online] [Cited: 08 14, 2008.] http://www.nda.ox.ac.uk/wfsa/html/u11/u1104_01.htm.

27. **Jubran, Amal.** Pulse Oximetry. *Intensive Care Med.* 2004.

28. **oxymetry.org.** [Online] 09 2002, 10. [Cited: 08 14, 2008.] http://www.oximetry.org/pulseox/principles.htm.

29. **Jubran, Amal.** Pulse Oximetry. *Critical Care.* 1999, Vol. 3.

30. **Klabunde, Richard E.** *Cardiovascular Physiology Concepts.* s.l. : Lippincott Williams & Wilkins, 2005.

31. **heartsite.com.** EKG or Electrocardiogram. [Online] [Cited: 08 14, 2008.] http://www.heartsite.com/html/ekg.html.

32. **Fogoros, Richard N.** The Electrocardiogram (ECG), What is it used for? [Online] 06 5, 2003. [Cited: 08 14, 2008.] http://heartdisease.about.com/cs/ekgecg/a/ECG.htm.

33. **Meditech.** CardioBlue looping ECG event recorder with Bluetooth. [Online] [Cited: 08 14, 2008.] http://www.meditech.hu/main.php?lang=en&page=techparam&device=cardioblue.

34. **World Book Encyclopedia.** *Temperature Body.* Chicago : Field Enterprises, 1996.

35. **Holtzclaw, BJ.** Monitoring body temperature. *AACN Clin Issues Crit Care Nurs.* 1993, Vol. 1.

36. **Mathie, M. J. and Lovell, N. H.** Determining Activity Using a Triaxial Accelerometer. *Engineering in Medicine and Biology.* 2002, Vol. 3.

37. **Bajcsy, Ruzena and Eklund, Mikael.** An Accelerometer Based Fall Detector: Development, Experimentation, and Analysis. *University of California.* 2005.

38. **Culhane, K. M. and O'Connor, M.** Accelerometers in rehabilitation medicine for older adults. *Age and Ageing.* 2005, Vol. 34.

39. **Bidargaddi, N. and Sarela, A.** Detecting walking activity in cardiac rehabilitation by using accelerometer. *Intelligent Sensors, Sensor Networks and Information.* 2007, Vol. 3.

40. *Comparison of the IEEE 802.11, 802.15.1, 802.15.4 and 802.15.6 wireless standards.* **Tjensvold, J. M.** 2007.

41. *802.15.1 IEEE Standard Part 15.1.* **IEEE Computer Society.** New York : IEEE, 2005.

42. *ZigBee.* **Bahl, Venkat.** s.l. : Philips, 2000.

43. **Bensky, Alan.** *Short-range Wireless Communication.* Burlington : Elsevier, 2004.

44. **Krasteva, R. and Boneva, A.** Application of Wireless Protocols Bluetooth and ZigBee in Telemetry System Development. *Problems of Engineering Cybernetics and Robotics.* 2005, Vol. 55.

45. **Spurgeon, Charles.** *Ethernet: The Definitive Guide.* s.l. : Elsevier, 2000.

46. **Otanez, P. and Parrott, T.** The Implications of Ethernet as a Control Network. *University of Michigan.*

47. *Inquérito à Utilização das Tecnologias da Informação e da Comunicação.* **Observatório da Sociedade da Informação e do Conhecimento.** 2004.

48. *802.11 IEEE Standard Part 11.* **IEEE Computer Society.** s.l. : IEEE, 2007.

49. *IEEE 802.11: Wireless LANs from a to n.* **Stallings, William.** s.l. : IEEE Computer Society, 2004.

50. **Crow, B. and Widjaja, I.** IEEE 802.11 Wireless Local Area Networks. *IEEE Communications Magazine.* 1997.

51. *GPRS Technology Overview.* **Sicher, A. and Heaton, R.** s.l. : Dell, 2002.

52. **Egea-Lopez, E. and Martinez-Sala, A.** Wireless communications deployment in industry: a review of issues, options and technologies. *Computers in Industry.* 2004, Vol. 56.

53. *PIC18F45J10 Family Data Sheet.* **Microchip.** 2007.

54. *PIC24HJXXXGPX06/X08/X10 Data Sheet.* **Microchip.** 2007.

55. *1 Mbit SPI Bus Serial EEPROM.* **Microchip.** 2006.

56. *ENC28J60 Data Sheet.* **Microchip.** 2004.

57. *AT45DB161D Flash Memory Data Sheet.* **Atmel.** 2008.

58. *128K SPI Bus Serial EEPROM.* **Microchip.** 2007.

59. *I2C Real Time Clock/Calendar Data Sheet.* **Intersil.** 2008.

60. *FT232R USB UART I.C.* **FTDI Chip.** 2005.

61. *The Microchip TCP/IP Stack.* **Microchip.** 2008.

ATTACHMENTS

A. Communication Protocols

Table: A.1: Detailed description of the 4 bytes that comprise one measure's data packet (25).

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	1	SNSD	OOT	LPRF	MPRF	ARTF	HR8	HR7
Byte 2	0	HR6	HR5	HR4	HR3	HR2	HR1	HR0
Byte 3	0	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Byte 4	0	R	R	R	SNSF	TURNON	CRITBAT	LOWBAT

SNSD:	Sensor Disconnect	An absence of signal, meaning the sensor is disconnected
OOT:	Out of Track	An absence of consecutive good pulse signals
LPRF:	Low Perfusion	Amplitude representation of low signal quality
MPRF:	Marginal Perfusion	Amplitude representation of medium signal quality
ARTF:	Artifact	Indicates an artifact condition
SNSF:	Sensor Fault	Sensor is providing unusable data for analysis
TURNON:	Turn On Mode	1 = Spot Check, 0 = Sensor
CRITBAT:	Critical Battery	Critical Battery condition
LOWBAT:	Low Battery	Low Battery condition
R:	Reserved	
HR8 – HR0:	Heart Rate	4-beat average displayed values including display holds
SP6 – SP0:	SpO_2	4-beat average displayed values including display holds

Table: A.2: Detailed description of Datacenter's request messages.

Name		Byte 1 Header	Byte 2 Length	Byte 3 Option
Connect_Req		1	0	-
Start_Req		2	0	-
Disconnect_Req	Option 1	4	1	0
	Option 2	4	1	1

Table: A.3: Detailed description of the Bluetooth socket's *Status* byte.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	R	R	R	R	OK	DATA AVAILABLE	PAIRED

Table: A.4: Detailed description of Transmission Module's response messages.

Name		Byte 1 Header	Byte 2 Length	Byte 3 Status
Connect_Resp	State1	1	1	1
	State2	1	1	0
Start_Resp		2	1	0
Disconnect_Resp		4	0	-

Table: A.5: Detailed description of the measure message.

Name	Header	Length	Stat	HR	S_PO_2	Sec	Min	Hr	Dt	Mth	Yr
Measure Message	3	9	-	-	-	-	-	-	-	-	-

Stat:	Status	Oximeter status byte
HR:	Heart Rate	Oximeter Heart Rate byte
SpO_2:	SpO_2	Oximeter SpO_2 byte
Sec:	Seconds	Measure's acquisition second
Min:	Artifact	Measure's acquisition minute
Hr:	Sensor Fault	Measure's acquisition hour
Dt:	Turn On Mode	Measure's acquisition day
Mth:	Critical Battery	Measure's acquisition month
Yr:	Low Battery	Measure's acquisition year

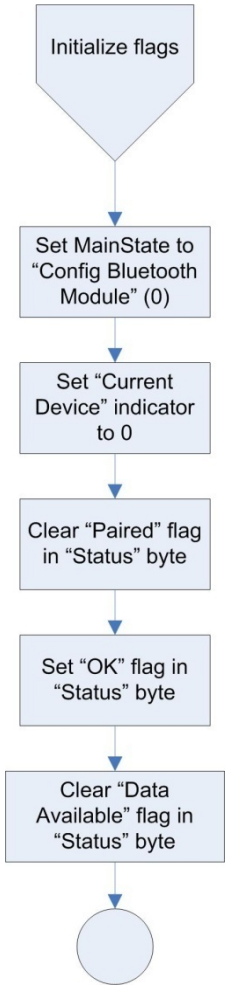
Table: A.6: Detailed description of the messages exchanged with Bluetooth Module (22).

Message	Details	Content (hexadecimal)	Length (Bytes)
Enter HCM	Enters Host Controlled Mode	01 04 FF 00 55 AA	6
Set OP mode	Sets Operating Mode	13 01 01	3
Config SP	Configures Serial Port	15 05 03 01 00 08 01	7
Run BT Module	Exit Host Controlled Mode	50 00	2
Scan Req	Scans for Bluetooth Devices	60 0C 00 9E 8B 33 05 04 00 00 00 00 00 00	14
Found Device (to Host)	Indication of Bluetooth Device found	61 0C 01 xx xx xx xx xx xx xx xx xx xx 00	14
End of Scan (to Host)	End of scanning process indication	61 01 00	3
Set Connect Rule	Set device to connect	17 06 xx xx xx xx xx xx	8
Set Security Mode	Sets the level of security of the connection	63 02 02 00	4
Set PIN Code	Sets PIN code of remote Device	65 06 35 30 31 36 37 30	8

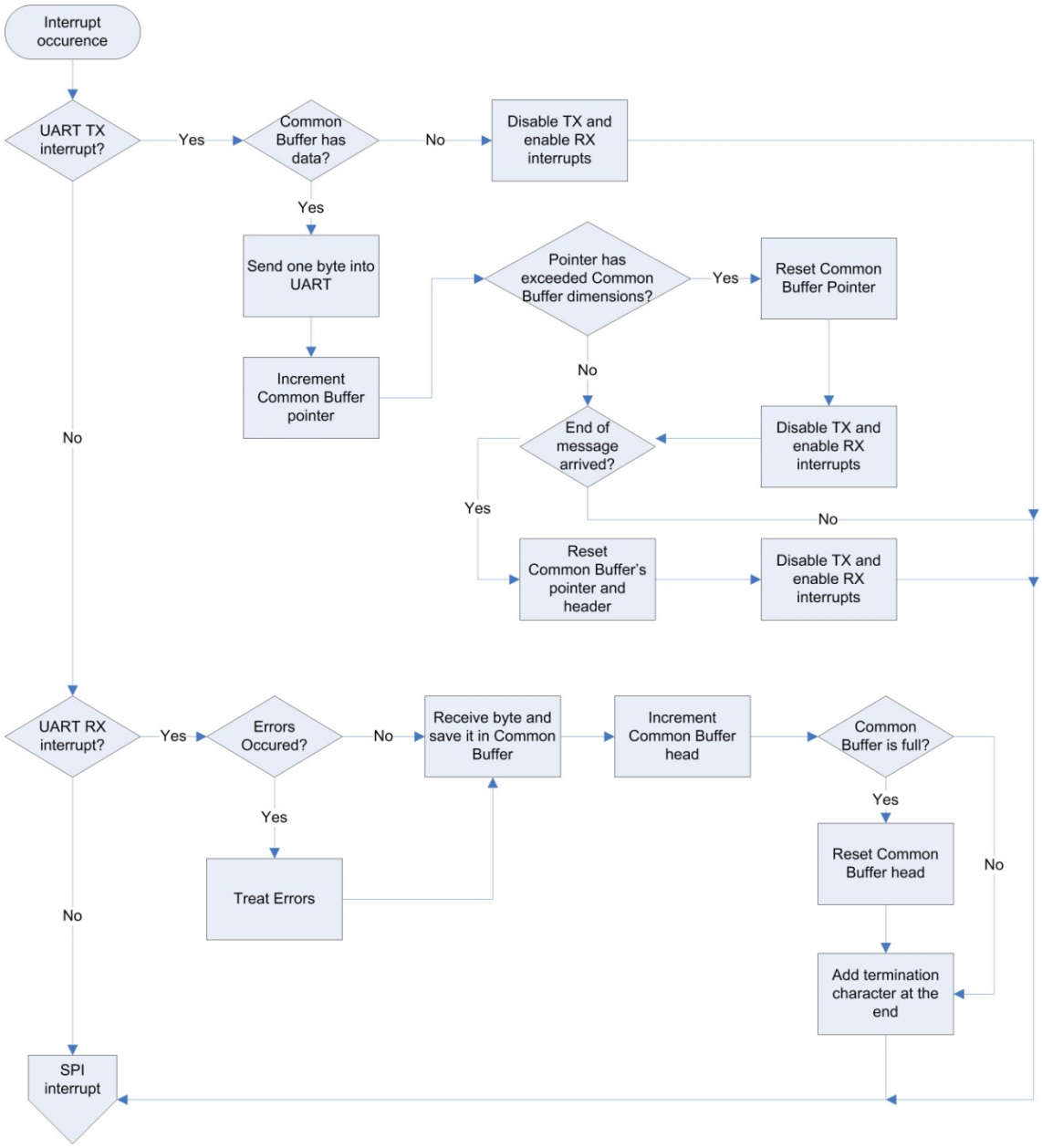
B. Flowcharts

B.1. **Bluetooth socket**

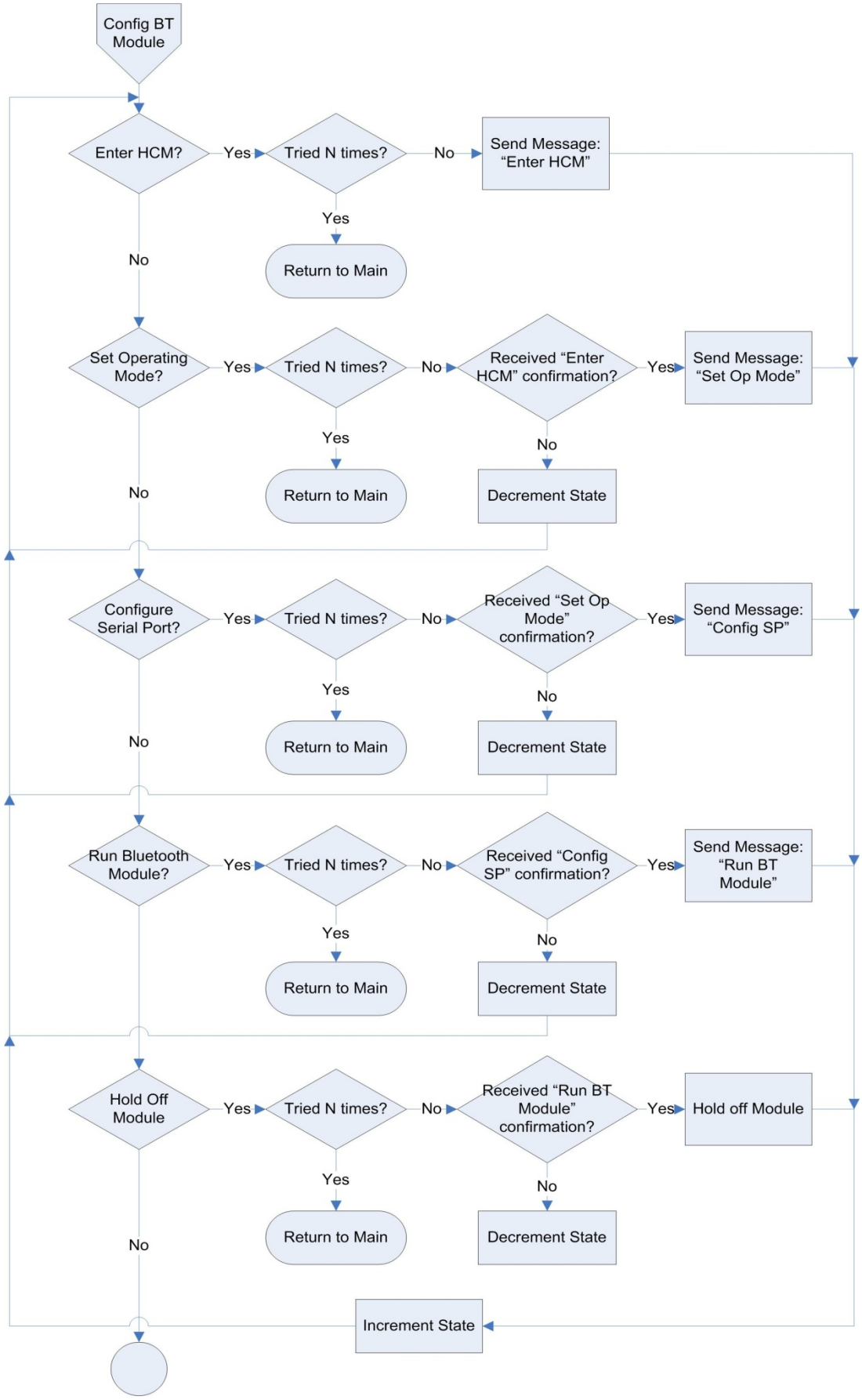
Initialize Flags



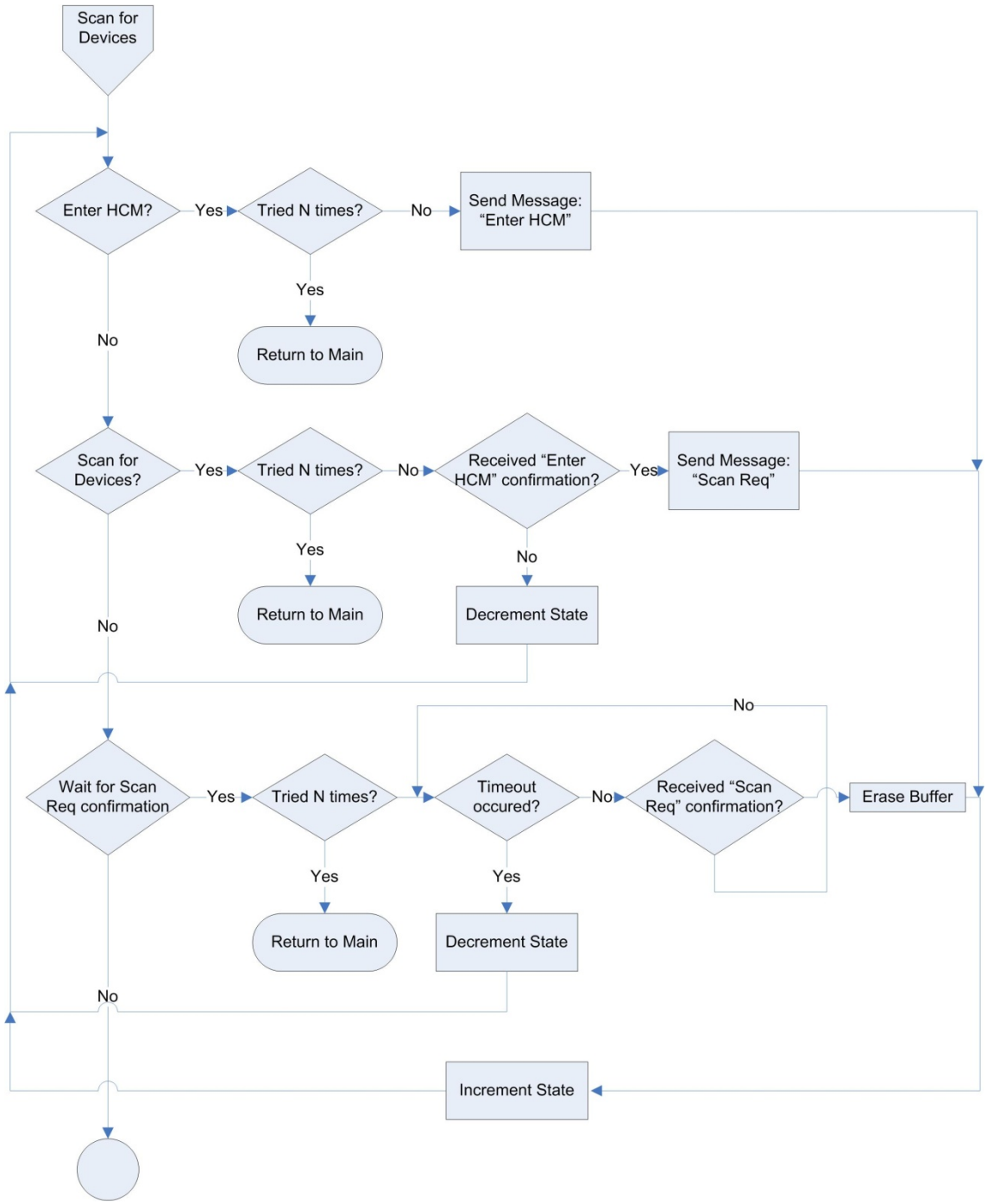
UART interrupts handling routines



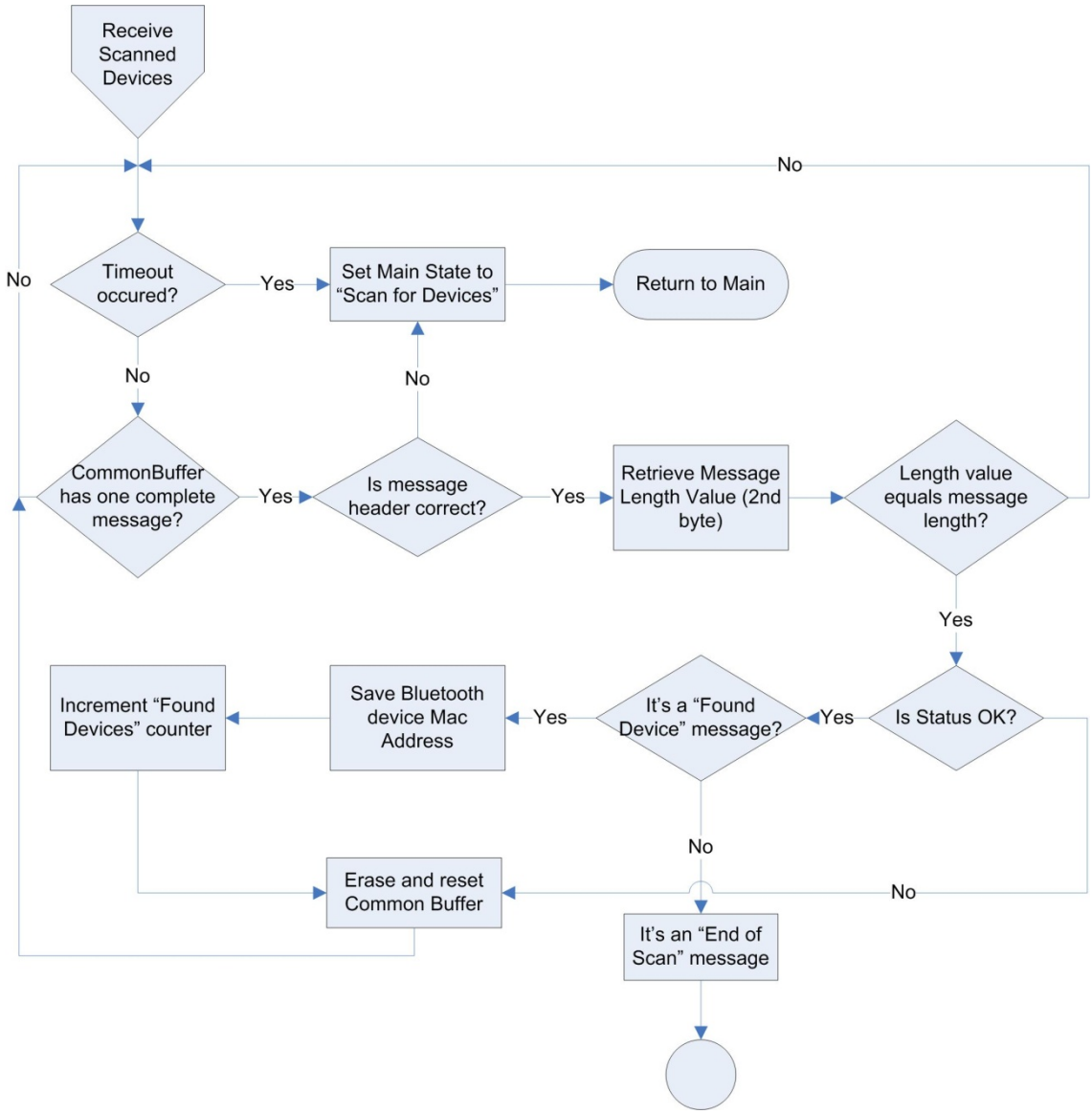
"Config BT Module" routine



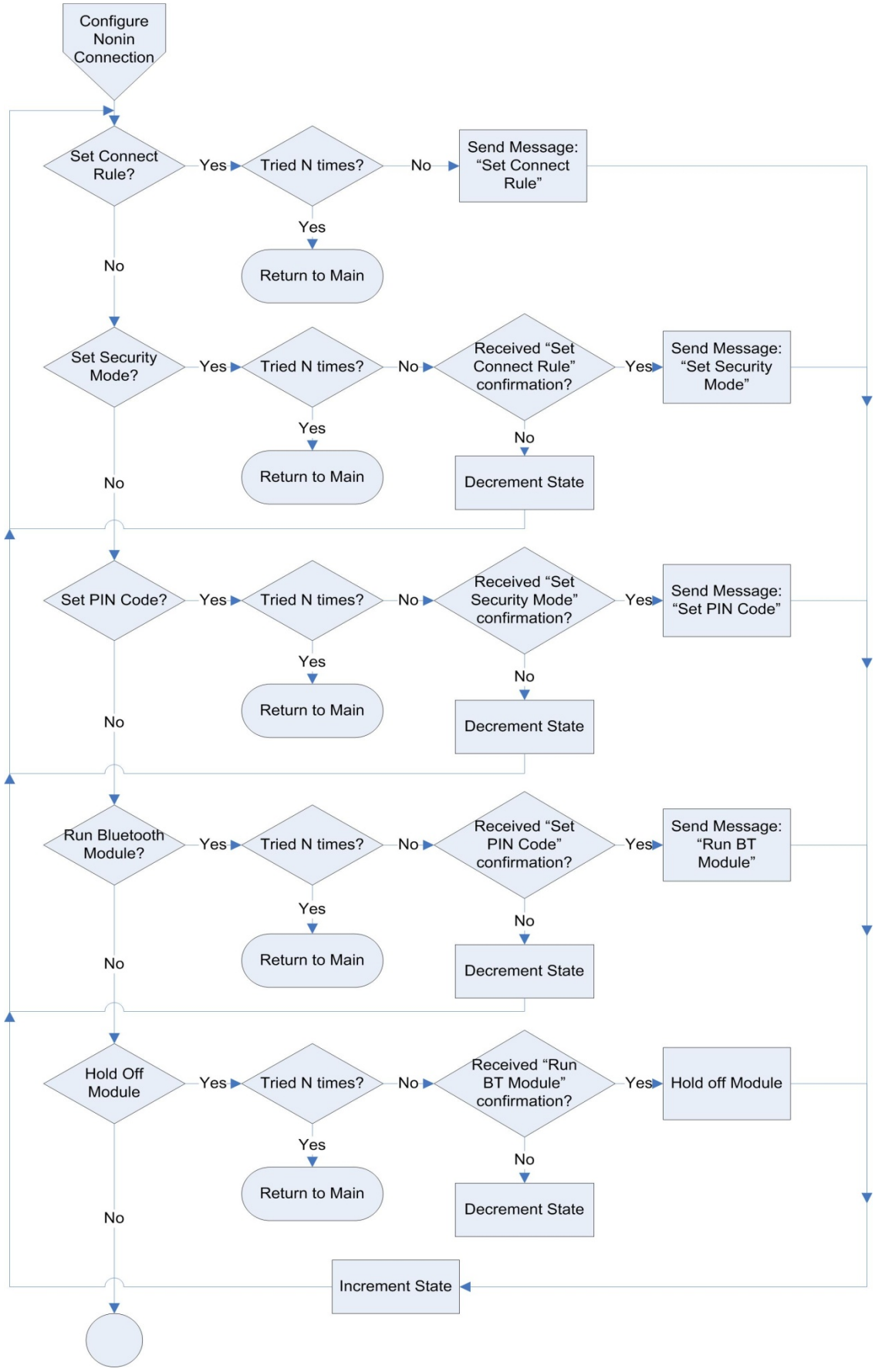
"Scan for Devices" routine



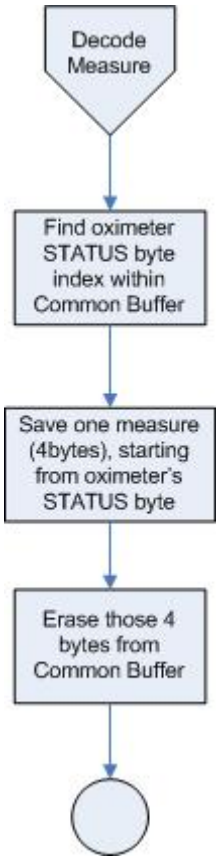
“Receive Scanned Devices” routine



"Configure Nonin Connection" routine

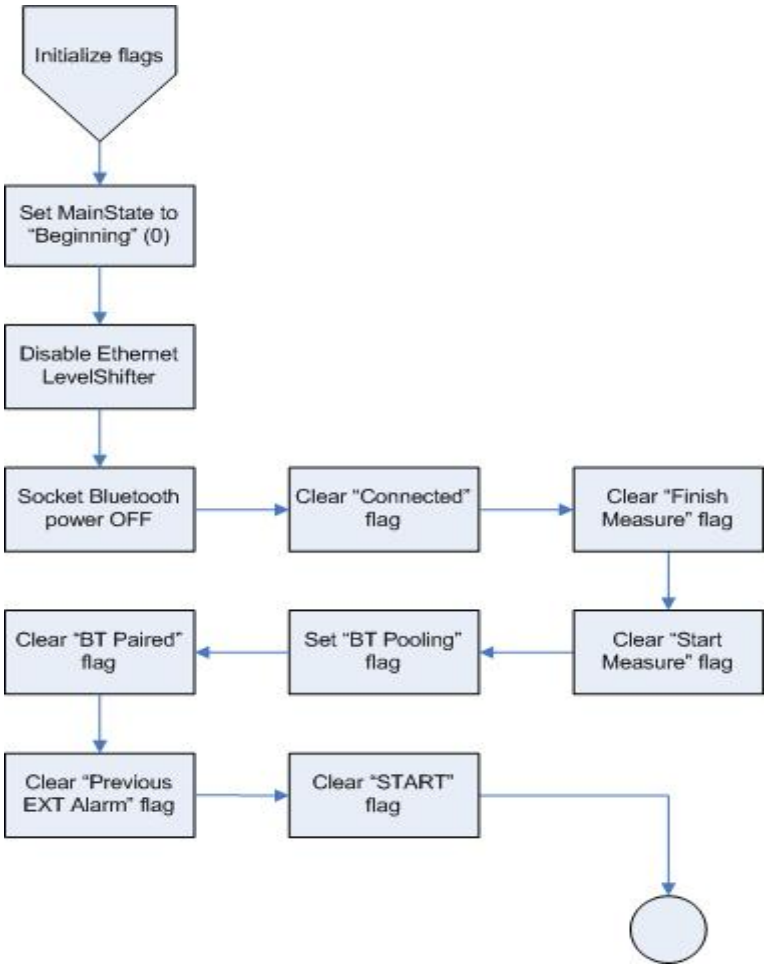


“Decode Measure” routine

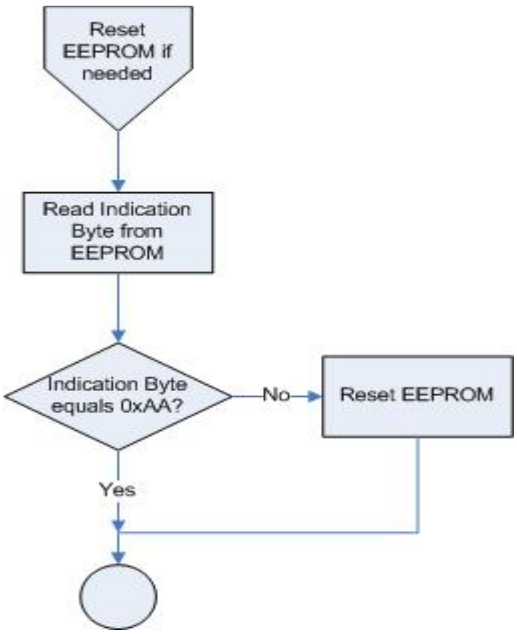


B.2. **Main Board**

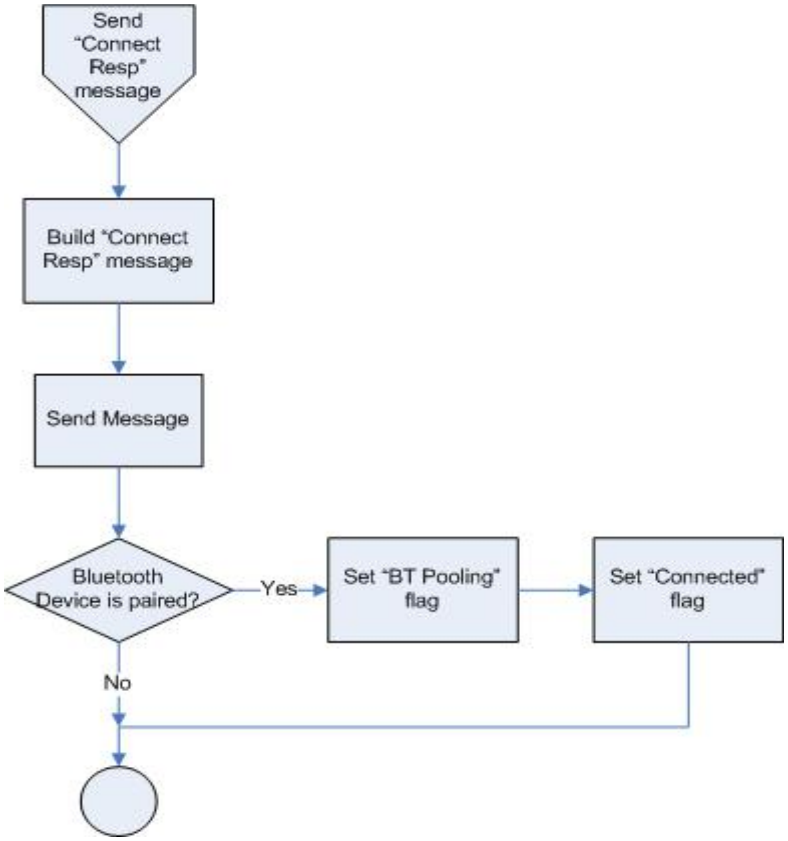
"Initialize Flags" routine



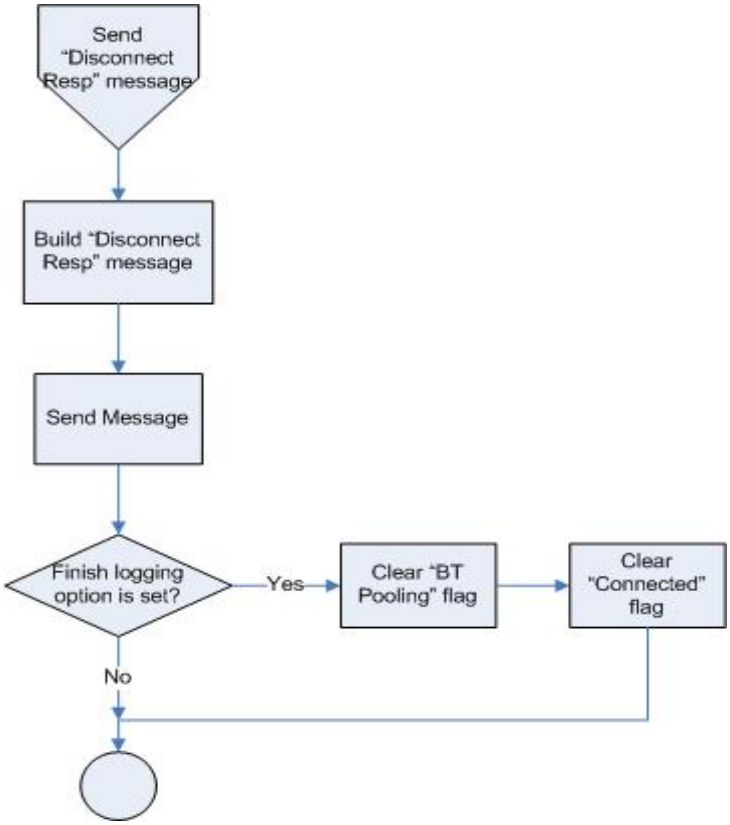
"Reset EEPROM" routine



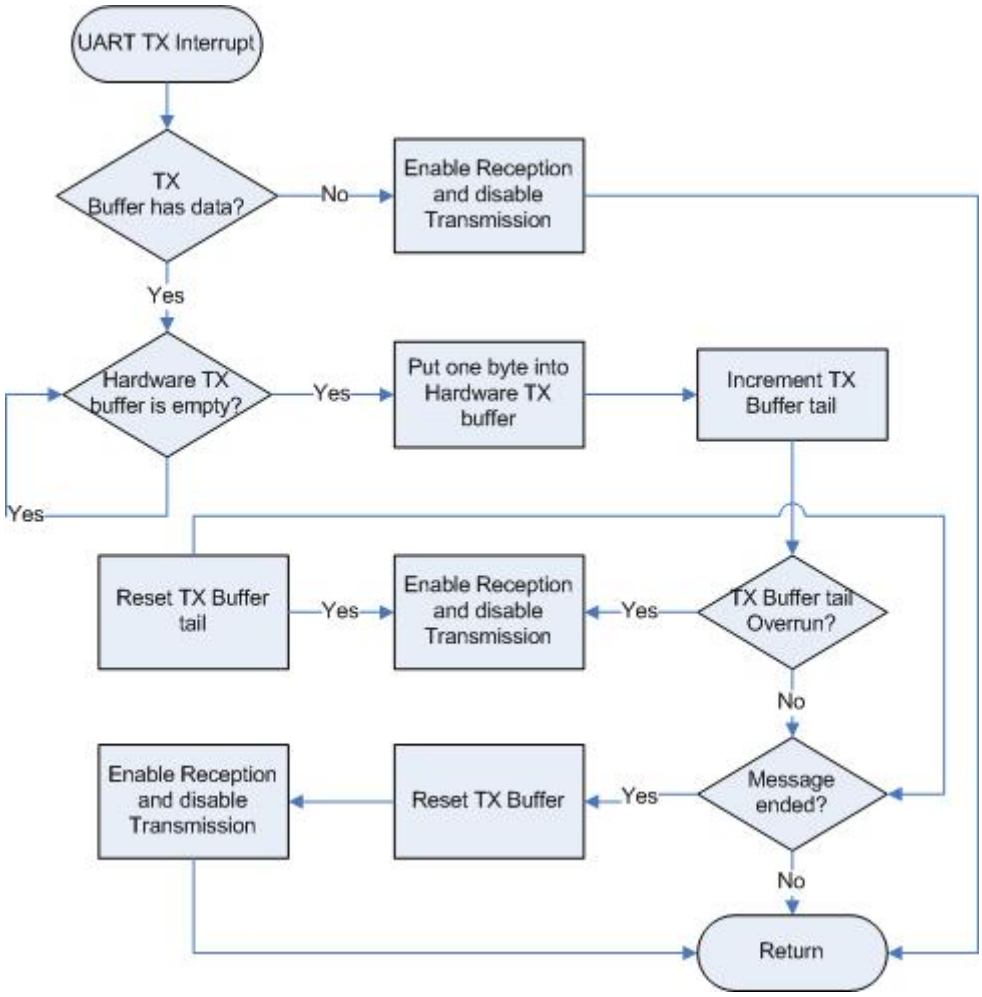
“Send Connect Resp” routine



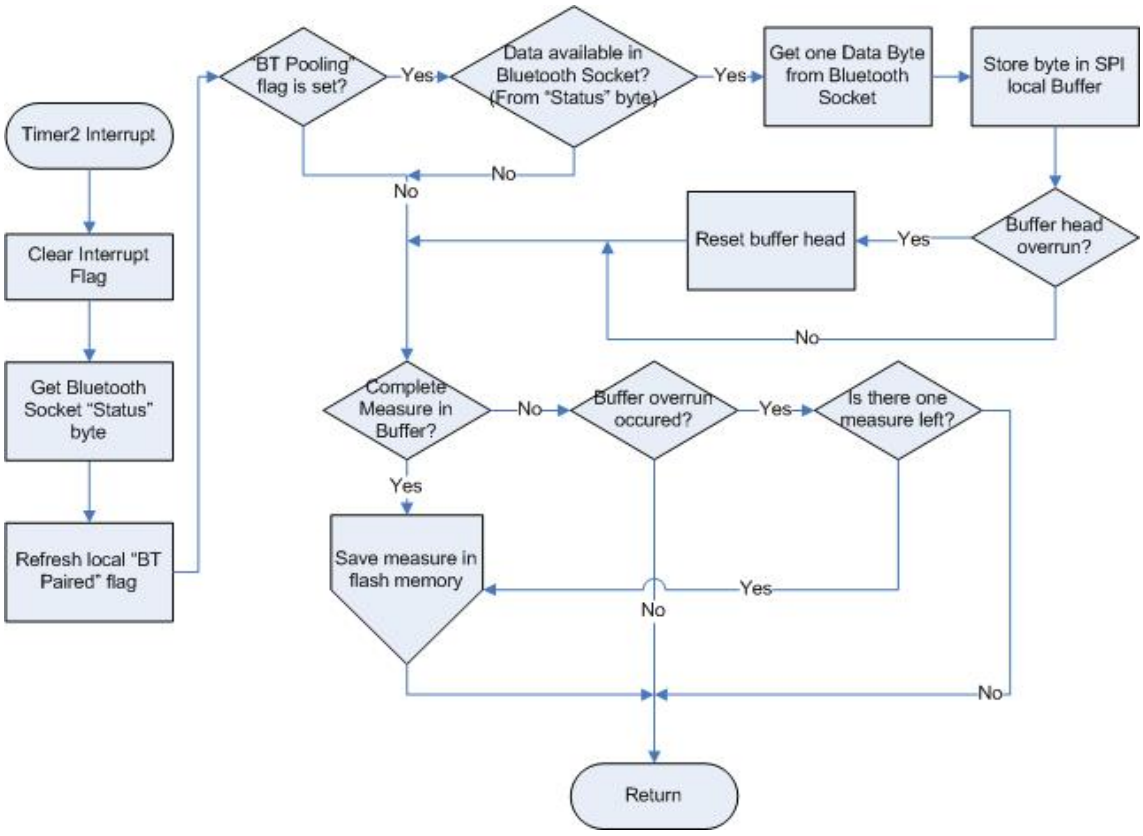
“Send Disconnect Resp” routine



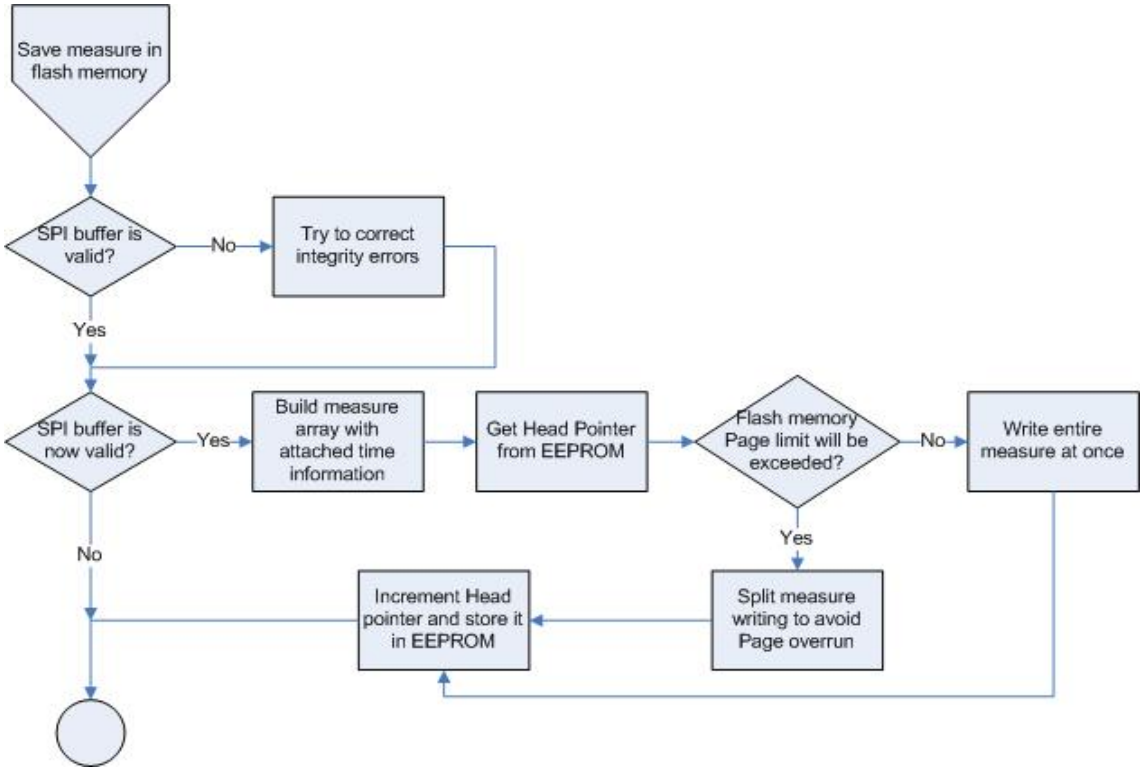
UART TX interrupt handling routine



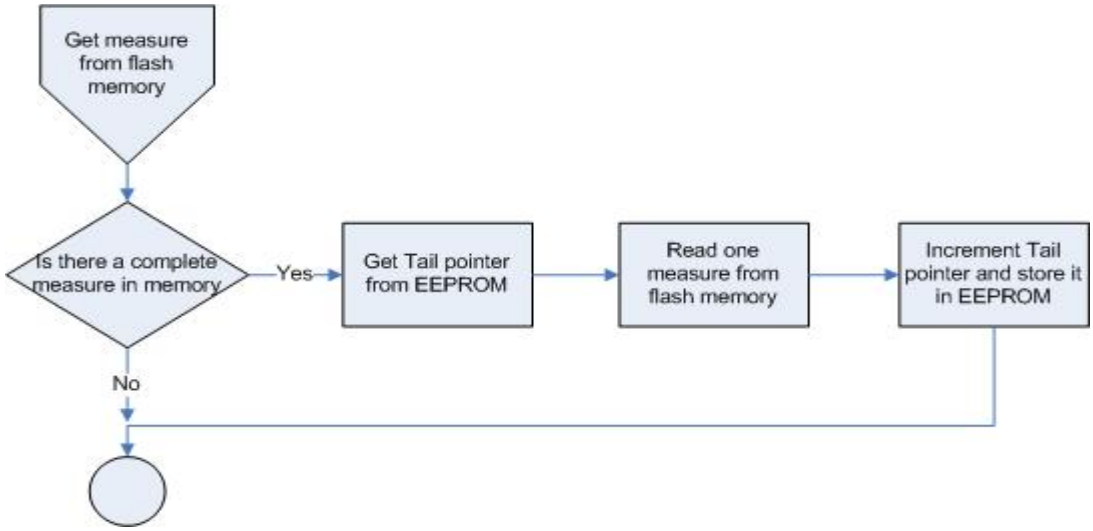
Timer 2 interrupt handling routine



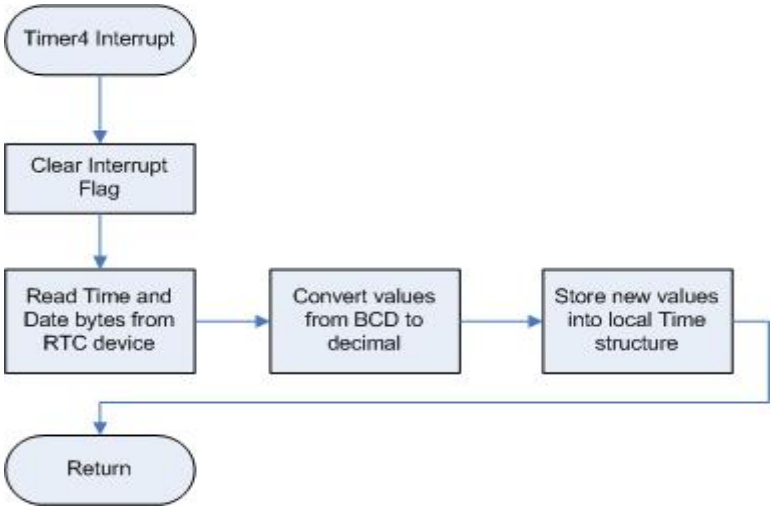
"Save measure in Flash memory" routine



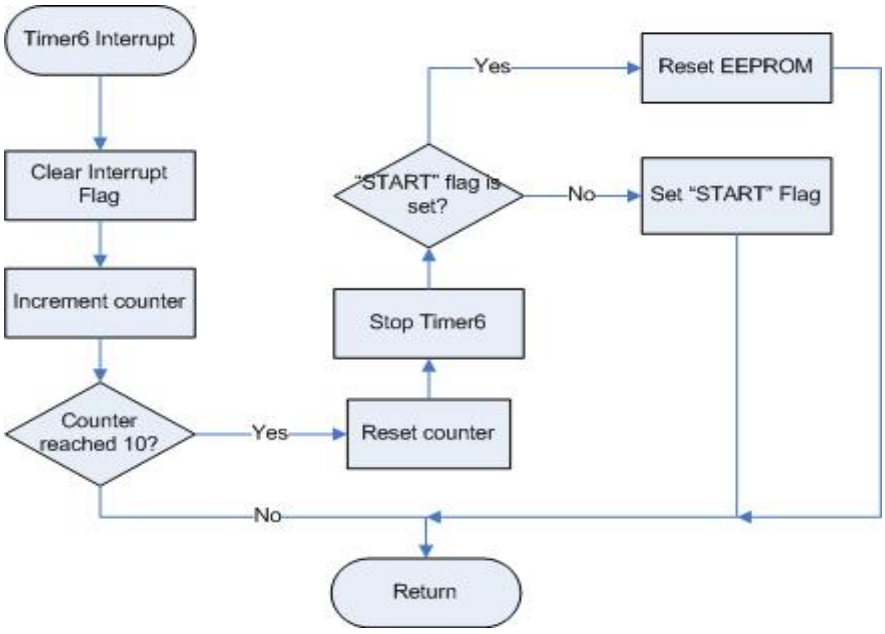
"Get measure from Flash memory" routine



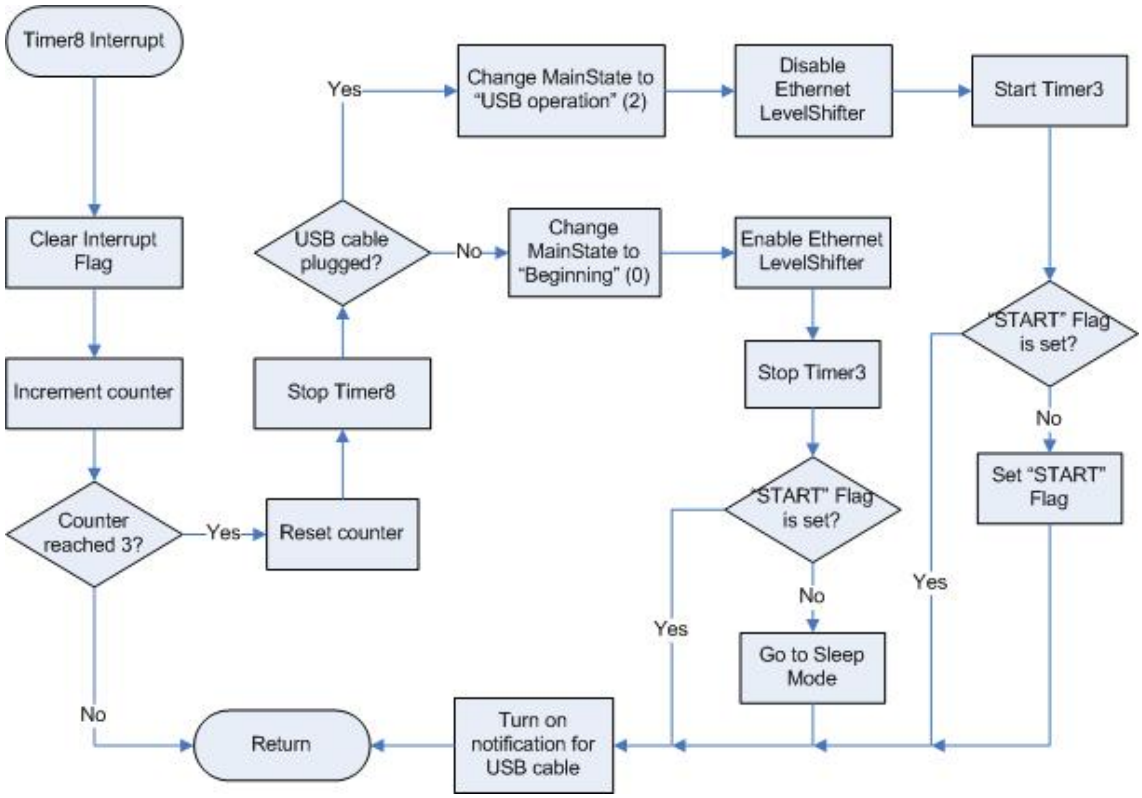
Timer 4 interrupt handling routine



Timer 6 interrupt handling routine



Timer 8 interrupt handling routine



C. Look4MyHealth Specification Document

ESPECIFICAÇÃO LOOK4MYHEALTH

SISTEMA DE RASTREIO DE DOENÇAS CRÓNICAS

Área de Tecnologia
Dep. Hardware
Confidencial
Versão 0.2

Índice

1. Introdução

A pressão arterial dos portugueses é demasiado alta. O Inquérito Nacional da Saúde 2005/06 mostra que a doença é comum a 20% da nossa população e, que só no último ano, 1,3% dos portugueses descobriram que são hipertensos, tornando-a a doença crónica com maior prevalência em Portugal.

Os especialistas em hipertensão consideram, porém, que os dados deste inquérito estão subestimados. Portugal é mesmo o país onde mais se morre e sofre por causa de acidentes vasculares cerebrais e, no entanto, dos dois milhões de portugueses que sabem que têm a pressão arterial descontrolada, estima-se que outros tantos tenham este problema e desconheçam o facto, segundo a Sociedade Portuguesa de Hipertensão.

O diagnóstico é feito através da medição da pressão arterial e pela verificação de que os seus níveis estão acima do limite normal. Contudo, um valor elevado isolado não é sinónimo de doença. Só é considerado hipertenso um indivíduo que tenha valores elevados em, pelo menos, três avaliações seriadas.

Um importante facto é ser possível prevenir a ocorrência de aproximadamente 40% dos acidentes vasculares cerebrais (AVC) e 25% dos enfartes do miocárdio caso a hipertensão arterial seja diagnosticada e controlada na devida altura.

A obesidade constitui também um dos maiores problemas de saúde nas sociedades ocidentais e é associada a problemas vasculares, hipertensão e diabetes. Quase 15% da população do nosso país é obesa e o excesso de peso afecta já 44,1% dos homens e 31,9% das mulheres portuguesas.

Para além do excesso de peso, a localização da gordura no corpo humano é um risco potencial para a saúde. Quando a gordura se encontra localizada principalmente na região abdominal ao redor da cintura, a propensão para desenvolver problemas de saúde, nomeadamente cardiovasculares, é maior do que quando a maior parte da gordura se situa na zona das coxas e dos quadris, ainda que o peso da pessoa seja considerado normal.

Com este projecto pretende-se construir um aparelho, que permita realizar uma medição medicamente fiável dos parâmetros anteriormente referidos, permitindo assim o tratamento numa fase precoce de uma condição de Hipertensão Arterial.

1.1. Objectivos

O objectivo do documento é definir as especificações para a criação do sistema Look4MyHealth. Com vista a esse objectivo será feita a descrição da arquitectura do sistema de forma a produzir linhas de orientação para o desenvolvimento deste. Para além disso, examinaremos os requisitos funcionais do sistema de modo a podermos contemplar toda a estrutura operante do mesmo.

1.2. Audiência

O documento destina-se às equipas dos departamentos de Engenharia de Software e de Sistemas Embutidos da ISA e a quaisquer serviços externos que sejam contratados para a realização do projecto.

1.3. Visão

O sistema Look4MyHealth destina-se às unidades clínicas e farmácias permitindo ao utilizador, para além da medição da tensão arterial, peso, medida da cintura abdominal, a sua identificação através de um sistema RFID. A informação medida será posteriormente armazenada, criando assim um histórico clínico do utilizador, podendo ser acedido posteriormente.

2. Arquitectura do Sistema

Globalmente, o Sistema Look4MyHealth será constituído por três domínios distintos:

- **Estação de Medição:** Será responsável pela medição dos parâmetros e pelo envio destes para a base de dados remota. Funcionará em farmácias e Unidades de Saúde Familiar. Esta estação comunica com a base de dados Remota via UMTS.

- **Base de Dados Remota:** Neste local ficarão armazenados todos os históricos dos utentes do serviço, que compreendem os dados dos parâmetros biológicos medidos e outras informações relativas ao utente. A base de dados concentrará as comunicações de todas as Estações de Medição e os acessos de cada Aplicação Web, disponibilizando a informação tanto às estações de medição (necessários quando efectuam um conjunto de medições a determinado utente) como aos utilizadores que acedem via Aplicação Web.

- **Aplicação Web:** Um portal Web permitirá acesso à informação de cada utilizador a partir de qualquer local com acesso à Internet. A aplicação possuirá duas funções distintas. A visualização do histórico das medições por parte do utente e a inserção de dados pessoais tais como o e-mail e número de telefone. A autenticação do utente na aplicação web é efectuada através de um User ID e de um PIN que são fornecidos aquando da primeira utilização.

2.1. Arquitectura Geral

A Arquitectura Geral explica globalmente o funcionamento do Sistema Look4MyHealth. Este permitirá a um utente efectuar medições dos três parâmetros biológicos referidos anteriormente (Pressão Arterial, Peso e Medida

da Cintura) e ter a possibilidade de estes serem armazenados remotamente para posterior acesso através da Aplicação Web.

O esquema seguinte mostra a arquitectura geral do sistema descrito.

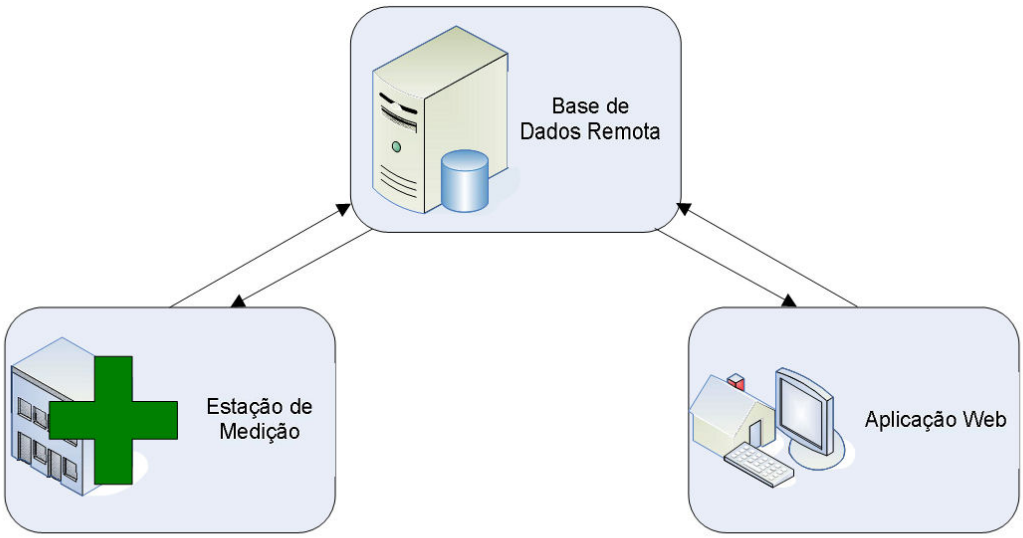


Figura 1 – Domínios do Sistema Look4MyHealth.

2.2. Arquitectura Física:

O esquema seguinte descreve a Arquitectura Física do Sistema Look4MyHealth:

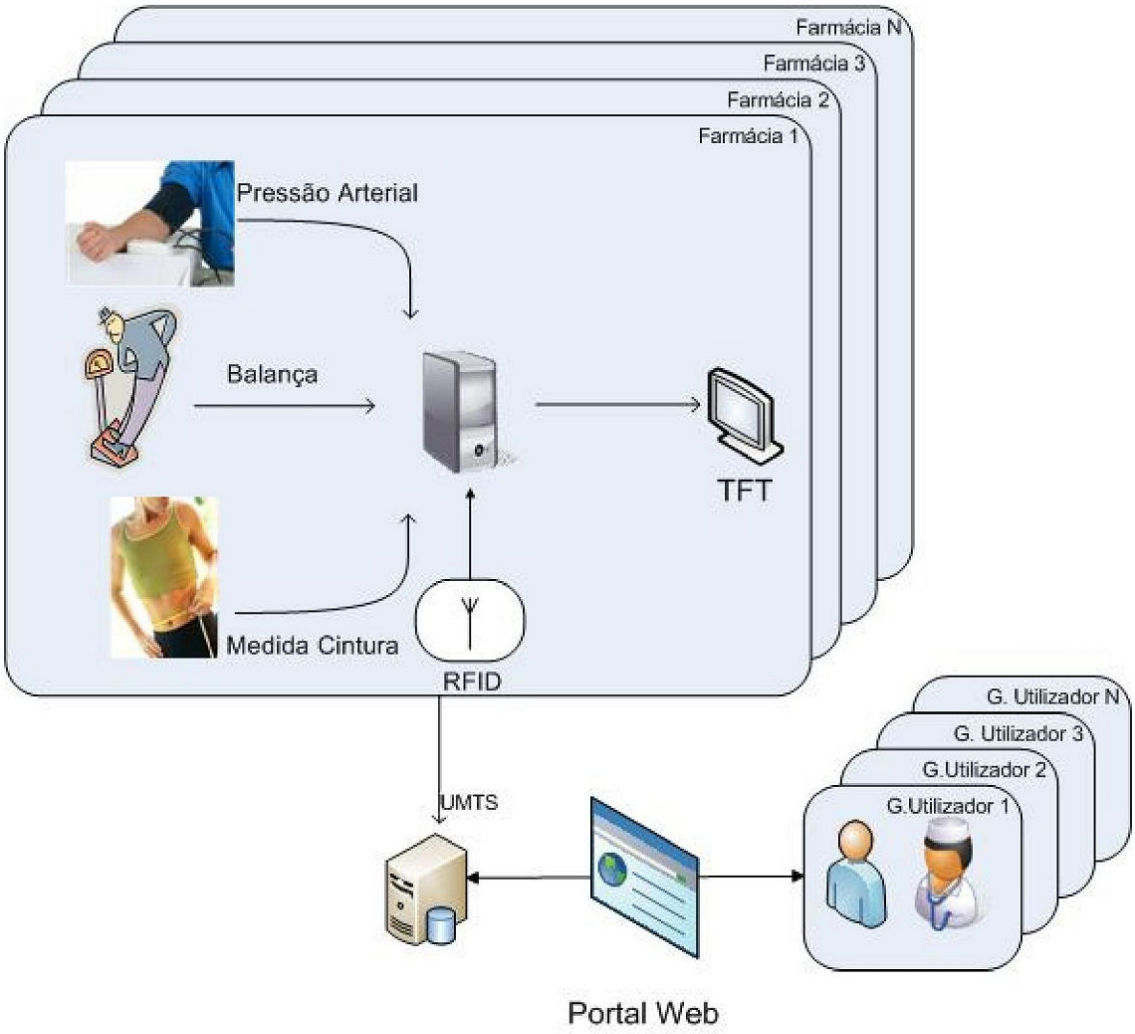


Figura 2 – Arquitectura Física do Sistema Look4MyHealth.

2.2.1. PC

O PC constitui a base de suporte para todos os outros componentes da Estação de Medição. Possui interfaces de ligação USB e série que providenciam ligação aos vários componentes da Estação (tanto os de medição como os de comunicação e de apresentação), bem como uma saída VGA para o display.

Decidiu-se pela utilização da *EdgeBox* da Critical Software, sendo que o sistema Operativo deste pacote é o Linux.

Este componente da Estação de Medição serve de suporte à aplicação de software que gere todos os componentes a ele ligados.

2.2.2. Medidor de Pressão Arterial

Este componente vai ser controlado por um módulo de software (ver Arquitectura Lógica) e os dados resultantes da medição serão transmitidos para o PC via interface RS-232.

O método de medição será o método oscilométrico (o mais indicado para os medidores de Pressão Arterial automáticos) que devolve a Pressão Sistólica, Pressão Diastólica e Pressão Média, para além da Taxa de Batimentos.

O medidor deverá estar ligado a uma braçadeira montada de forma a se poder alternar facilmente entre o braço esquerdo e direito. Esta arquitectura será necessária visto que para uma medição mais precisa é necessário saber em que braço a pressão tem valores mais elevados.

2.2.3. Balança

Este componente é também controlado por um módulo de software, sendo que os dados são transmitidos para o PC por USB/RS-232 no final de cada medição.

2.2.4. Fita de Medição de Cintura

Visto que não existe no mercado uma solução que vai de encontro a todos os requerimentos será necessário construir um sistema que faça o que é pretendido. O sistema basear-se-á numa fita ou num fio em torno de um carreto com um imane que, a cada volta, gera um impulso eléctrico induzido, para que a electrónica conte o número de impulsos. Existe ainda a possibilidade de conceber o sistema com conectividade wireless, o que aumentaria o conforto de utilização.

2.2.5. Apresentação dos Dados

2.2.5.1. Display

Este componente de apresentação tem a função de apresentar imagens e mensagens elucidativas ao utilizador de forma a guia-lo no processo que compreende a medição dos três parâmetros. Serve ainda para apresentar ao utilizador questionários. Este componente da Estação de Medição é controlado por software de forma a ser possível apresentar as mensagens de forma sincronizada com o processo de medição. Possui ainda uma interface de utilização táctil o que possibilita a interacção entre a estação de medição e o utilizador, o que torna o processo *user-friendly*.

2.2.6. Módulo RFID

Na altura da medição cada utente vai ter que possuir uma Tag RFID que torna possível a sua identificação perante a base de dados. O módulo RFID vai permitir a leitura desta Tag e comunica-la ao PC para que o software possa identificar o utente junto da Base de Dados Remota.

2.2.7. Módulo UMTS

O módulo UMTS vai permitir o estabelecimento de uma ligação entre a Estação de Medição e a Base de Dados Remota. Esta ligação será feita através da Internet. Depois de estabelecida a ligação torna-se possível tanto o envio dos dados relativos ao processo de medição como a consulta, tanto dos dados de um utente, necessários à realização deste processo como de outras informações presentes no servidor remoto. A identificação do utente junto da Base de Dados é feita através da sua Tag RFID.

2.3. Arquitectura Lógica:

A figura seguinte ilustra os módulos existentes no Sistema de Medição, bem como as relações entre estes. As setas a azul representam uma interface de ligação física (USB, RS-232 e VGA), enquanto a seta a cinzento representa a relação entre os módulos de software apoiados no Computador e que constituem a Aplicação.

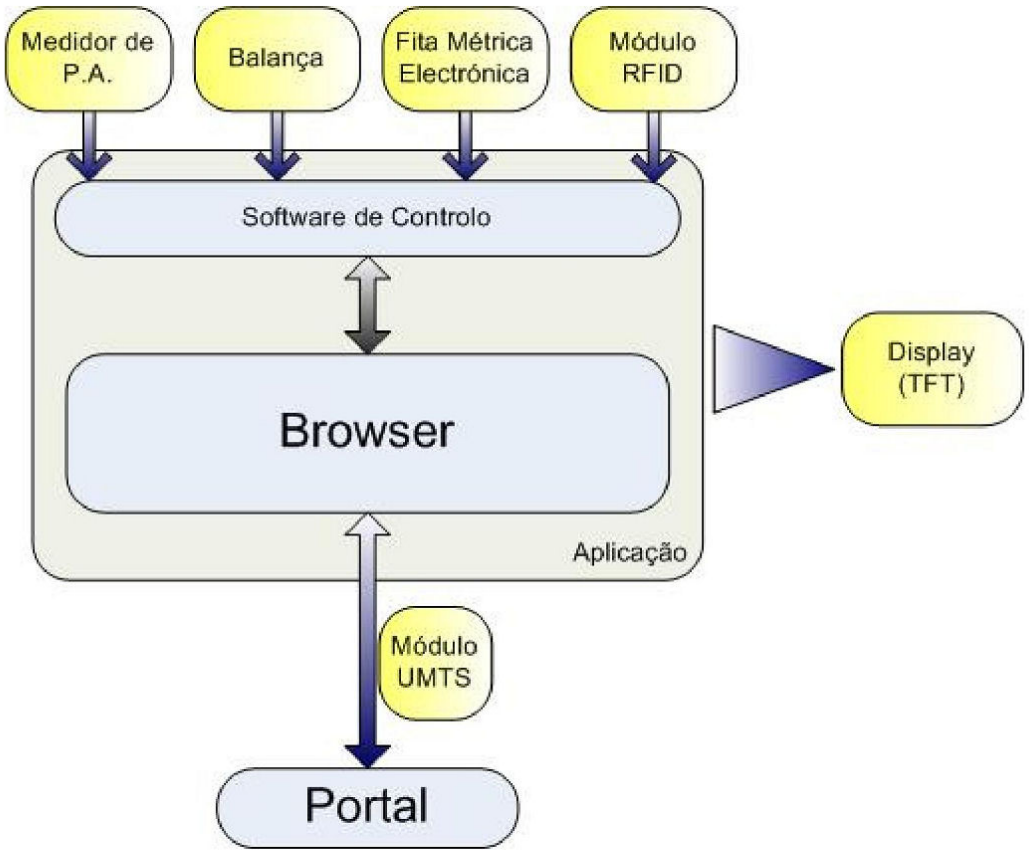


Figura 3 – Arquitectura Lógica da Estação de Medição

2.3.1. Software de Controlo:

Este módulo de software vai permitir comunicar com os diversos dispositivos de medição (medidor de P.A., Balança e Fita Métrica Electrónica), bem como com o módulo de RFID, com rotinas que os permitam controlar e que, após finda a comunicação, permitam receber adequadamente os dados. Este módulo de software deverá ter a capacidade de controlar o fluxo das medições que vão sendo efectuadas ao longo do processo, ou seja o utilizador

poderá dirigir-se por qualquer ordem aos aparelhos de medição visto que a aplicação detecta a sua presença. Deste modo é ainda possível repetir medições.

2.3.2. Browser

Este módulo da Aplicação é o responsável por mostrar informações provenientes do Servidor Remoto. Estas informações compreendem os questionários ao utente.

2.3.3. Aplicação

A aplicação será um módulo de software local que estará instalado no PC da estação de medição e suportará os outros módulos de software. Será a Aplicação a responsável pelo controlo a alto nível do processo.

3. Utilizadores

Existem diversos tipos de utilizadores que se agrupam pelo grau de permissão que possuem ao sistema. Os diversos grupos são:

- Administrador (A)
- Farmacêutico (F)
- Utente (U)

3.1. Permissões

Tendo em conta a Aplicação Web, o papel de cada grupo de Utilizadores é o seguinte:

Tabela 1 – Permissões concedidas a cada um dos tipos de Utilizadores

Funcionalidades	A	U
Adicionar/Eliminar Equipamentos	✓	
Criar/Alterar/Eliminar Contas de F	✓	
Criar/Alterar/Eliminar Contas de U	✓	
Aceder Historial Clínico		✓
Aceder/Alterar Dados Pessoais		✓

Legenda: A – Administrador; **F** – Farmacêutico; **U** – Utente

3.2. Hierarquia

Desta forma poderemos definir a hierarquia do sistema, no que diz respeito à Aplicação Web:

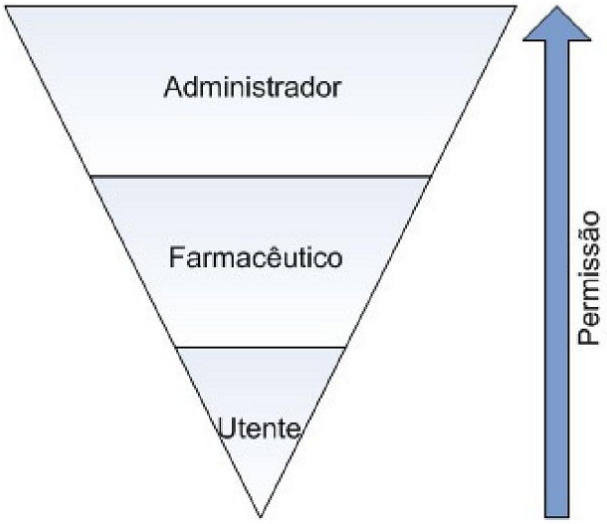


Figura 4 – Hierarquia dos utilizadores

4. Fluxo de Utilização

4.1. Autenticação

São entregues vários pacotes à farmácia que contêm um cartão RFID para identificação junto da Estação de Medição, bem como um User ID e um PIN, para aceder à Aplicação Web. Antes da primeira utilização, cada utente deverá possuir um desses pacotes que deverá usar para autenticação. No caso da Estação de Medição a autenticação será feita apenas por aproximação do cartão RFID à zona de detecção. No caso da Aplicação Web, existirá uma página de login onde serão introduzidos os dados identificativos desse utente.

4.2. Medição dos parâmetros

O fluxo de processo de medição dos parâmetros é aleatório, ou seja, após autenticação o utente pode efectuar qualquer medição, podendo até repetir medições. O sistema detectará quando o utente se encontra a utilizar um determinado módulo de medição.

4.2.1. Medição do peso

Logo que o Sistema detecte uma alteração no valor do peso dado pela balança assumirá que o utente irá efectuar esta medida. Assim que o valor esteja estabilizado durante poucos segundos o Sistema fixa esse valor, grava-o e mostra-o no display.

4.2.2. Medição da Cintura

Tal como na medição do peso, assim que haja uma alteração no valor dado pela fita de electrónica de medição o sistema assumirá que o utente irá efectuar esta medida. O final da medição será detectado

quando o circuito da fita for fechado. Finda a medição, o valor será gravado e mostrado no display.

4.2.3. Medição da Pressão Arterial

O módulo de pressão arterial tornar-se-á activo quando o paciente se sentar na cadeira, devido ao sensor de movimento que esta incorpora. De modo a que o resultado da medição possa ser considerado medicamente aceitável o utente necessita de estar 10 minutos em repouso completo. Porém, esse tempo de espera não é obrigatório, sendo da opção do utente decidir se espera ou não. Durante esse período de tempo irão ser mostradas algumas imagens no display, com carácter de entretenimento, cujo conteúdo depende se a pessoa utiliza o aparelho pela primeira vez ou não.

4.2.3.1. Primeira Utilização

Quando o utente utiliza o aparelho pela primeira vez é lhe pedido que insira alguns dados pessoais. Para além disso, elucida-se o paciente que se irá medir a pressão nos dois braços, para saber qual das pressões é a mais elevada.

4.2.3.2. Não é a Primeira Utilização

Se o utente já tiver utilizado o aparelho, serão mostrados, ao longo dos 10 minutos, alguns inquéritos ou vídeos. O preenchimento dos inquéritos é de carácter facultativo.